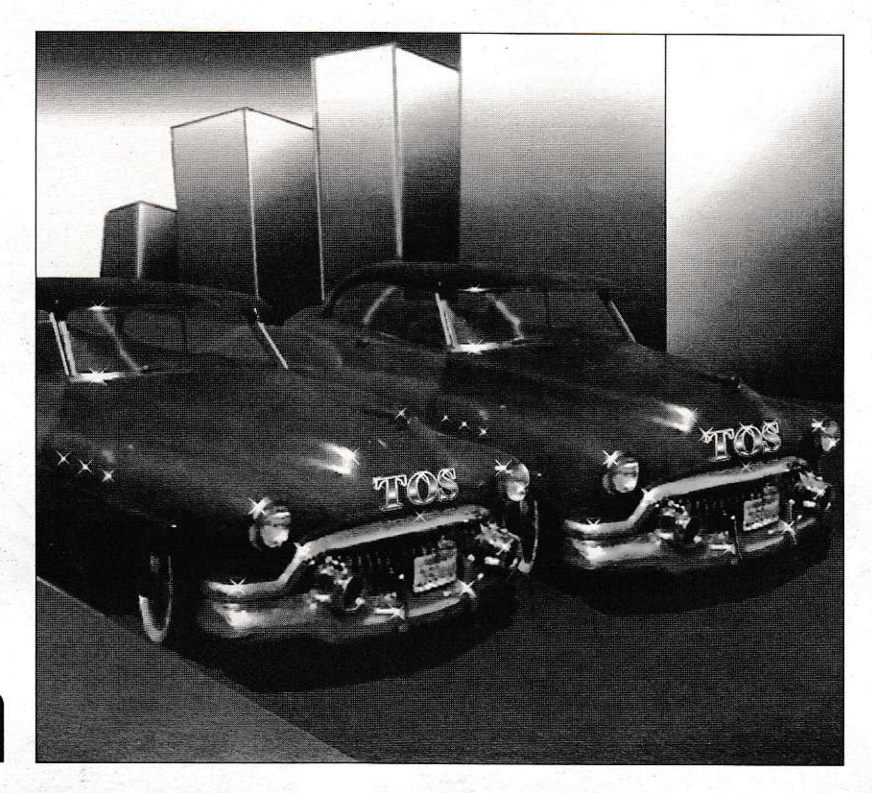

ATARI
ST COMPUTER
Die Fachzeitschrift für ATARI ST- und TT-Anwender
Dezember 91
DM 8,-
Ös. 64,-
Sfr. 8,-
Lit. 7500,-
12
MIDI-Software
• Freestyle
• Session Partner
Software:
• Repro Studio Pro
• Type Art
• Querdruck 2
• Music Mon
AHDI
selbst
konfiguriert
I/O-Port
im Eigenbau

EDITORIAL

New TOS in Town



Es ist mal wieder soweit. Alle Atari-Besitzer können sich neuer TOS-Versionen erfreuen. Dabei sind diesmal sogar ST- als auch TT-Besitzer berücksichtigt, da es für beide Rechnertypen ein TOS-Update gibt. TOS 2.06 (ST) und TOS 3.06 (TT) lauten die Versionsnummern, der wohl in Kürze erhältlichen Versionen, d.h. in den neuen TTs findet man wohl schon das neue TOS zusammen mit einem HD-Laufwerk und einem Atari-eigenen Floppy-Controller. Ansonsten findet man neben den Vorzügen des neuen Desktops, die bisher nur TTs und STEs gegönnt waren, noch einige interessante Neuerungen. Mehr dazu können Sie in den News dieser Ausgabe lesen. Probleme gab es bei den STs, da das neue TOS 2.06 aufgrund der Anzahl der Pins der ROMs eigentlich nur für die STE-Serie bestimmt war. Hier schafft eine Platine eines Fremdanbieters Abhilfe, durch die man auch ST-Modelle ohne das „E“ auf den neuesten Stand bringen kann. Zwar kann man mittlerweile auch einen Software-Patch für TOS 2.05 (also eine Versionsnummer kleiner) in einigen Mailboxen finden, damit diese Version auch auf alten Rechnern läuft; doch stellt sich hier mal wieder die Frage, inwieweit eine solche Lösung die Copyright-Ansprüche Ataris verletzt. Allerdings dürfte eine Hardware-Lösung immer vorzuziehen sein, auch wenn sie, wie in diesem Falle, DM 198,- kostet.

Harald Egel

INHALT

SOFTWARE

HARDWARE

GRUNDLAGEN



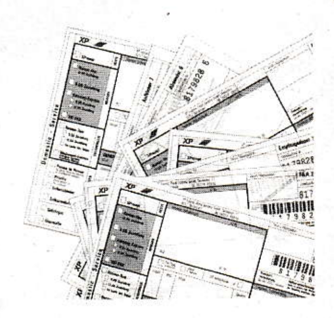
MIDI

Freestyle & Session Partner

In unserem MIDI-Schwerpunkt wollen wir Ihnen neben ein paar Grundlagen zwei Programme vorstellen: Freestyle und Session Partner. Die Programme haben sehr viel gemeinsam, sind aber dennoch völlig verschieden. Freestyle ist eher als Begleitautomat/ Arrangierer anzusehen und Session Partner als eine Art Groove-Designer mit umfangreichen Zufallsfunktionen. Beide benötigen externe Klangquellen wie Keyboards oder Expander. Wir wollen beide Programme nicht miteinander vergleichen, sondern beide unabhängig voneinander beleuchten. Lesen Sie dazu mehr ab

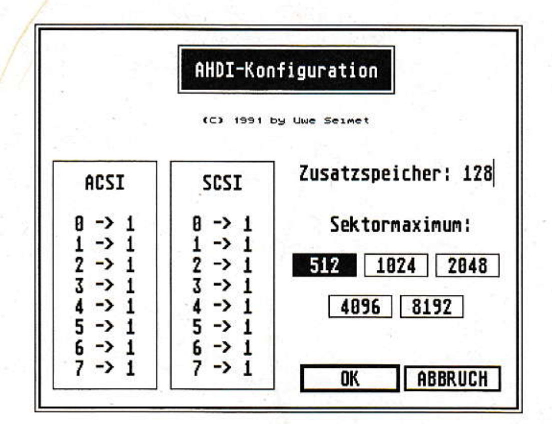
Seite 20

Formular-programme



Unter dem Motto "Kampf dem Papierwust" wollen wir Ihnen vier Programme vorstellen, die einem die lästige Arbeit des Formularausfüllens erleichtern sollen. Die große Volkszählung ist zwar schon lange vorbei, aber wer hat nicht Banküberweisungen u.ä. Formulare häufig vor sich liegen. Ob der Atari eine große Hilfe beim Ausfüllen ist oder ob man besser einen Stift zur Hand nimmt, versuchen wir in unserem Artikel zu beantworten. Merken Sie sich auf Ihrem Muß-ich-unbedingt-lesen-Formular:

Seite 43



AHDI-Konfiguration

Bereits seit dem Spätsommer 1989 liegt der Atari-Festplattentreiber AHDI in der Version 3.0 vor, die inzwischen durch die Version 4.0, die in erster Linie für die Besitzer des Atari TT interessant ist, ersetzt wurde. Eine besondere Eigenschaft beider Treiber ist der Umstand, daß diese konfiguriert werden können. Leider gehört ein geeignetes Konfigurationsprogramm nicht zum Lieferumfang. Grund genug, selber ein solches Programm zu schreiben.

Seite 162

Repro Studio pro

Fristeten frühere Versionen des REPRO STUDIO ST eher das Dasein eines Mauerblümchens, so bringt Trade It ihr Programm nun in einer 'Pro' (für Professional)-Version mit erheblich erweitertem Funktionsumfang auf den Markt. Inwieweit es Trade It gelungen ist, mit ihrem Programm in die Spitze der Bildverarbeiter auf dem Atari ST/TT vorzustoßen und ob das Kürzel 'Pro' die Interpretation 'PROfessionelles PROgramm' oder eher 'PROblematisches PROdukt' verdient, zeigt unser Test. Soviel vorweg: Diejenigen unter uns, die schon frühere Versionen des Repro Studios kennen, werden angenehm überrascht sein.

Seite 30



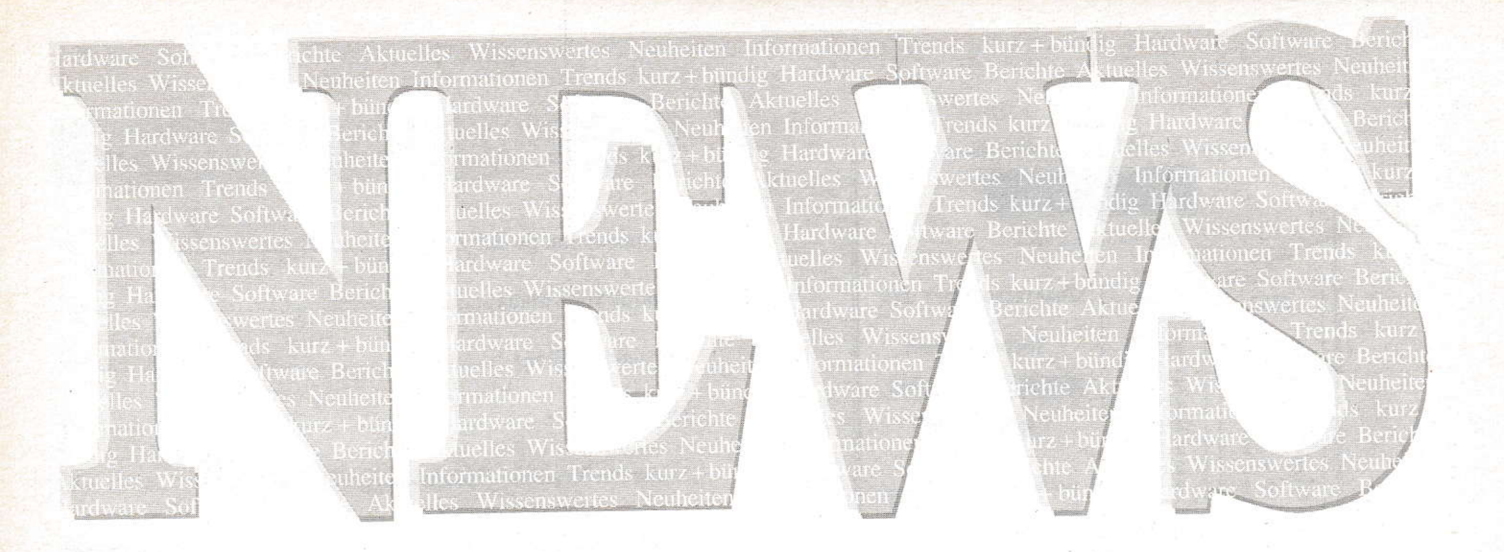
PROGRAMMIERPRAXIS

PUBLIC DOMAIN

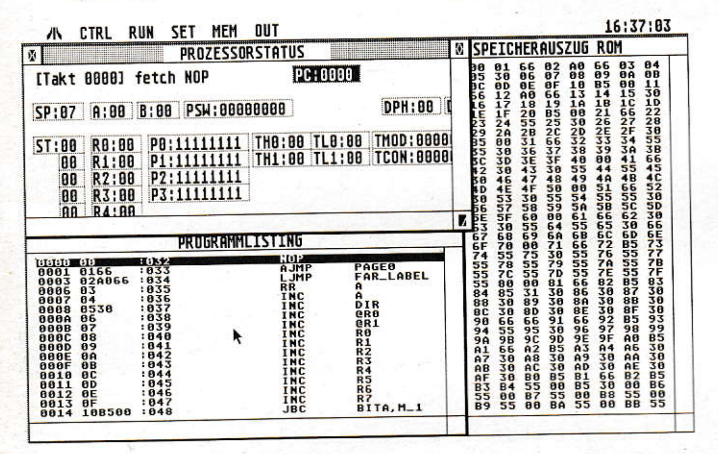
ST-REPORT

AKTUELLES

RUBRIKEN



uC31/51-ST. Die Entwicklungsumgebung.



Mit der uC31/51-ST-Entwicklungsumgebung existiert für den ATARI ST eine vollständige Software-Entwicklungsumgebung für die Micro-Controller der 80xx(x)-Familie in der Version 2.00. Sie ist ausgestattet mit einem Makro-Assembler, Disassembler, Formatkonverter und mit einem Simulator. Die Bedienung dieses Komplettpaketes ist voll GEM-gesteuert.

Der 2-Pass-Assembler erzeugt direkt ohne Linker Objekt-Code, der mit dem Disassembler wieder rückverarbeitbar ist. Der Formatkonverter erzeugt sowohl aus dem Objekt-Code des Assemblers ladbare Files für alle gängigen EPROMMER (auch den Juniorprommer von MAXON) als auch rückwärts aus den EPROMMER-Files wieder Objekt-Code. Damit steht eine Schnittstelle zu anderen Entwicklungssystemen zur Verfügung. Mit dem Simulator werden Laufzeitfehler entdeckt. Sein Aussehen gleicht mit seinen drei Fenstern dem von HLL-Debuggern. Im ersten Fenster wird der Inhalt der Prozessorregister, wahlweise dez., hex. oder binär angezeigt, im zweiten wird das Listing mit Markierung von aktueller Programmzeile und Breakpoints und im dritten der Speicherinhalt angezeigt.

Abgerundet wird die Entwicklungsumgebung durch einen Makro-Rekorder für Tastatur- und Menübefehle. Die Entwicklungsumgebung ist erhältlich für die Controller 8031/51, 8032/52, 80515/535, 80517/537.

Der Preis für die GEM-Version inkl. Handbuch beträgt DM 298,-. Assembler, Disassembler und Formatkonvertierer allein als TTP-Files sind inkl. Handbuch für DM 149,- erhältlich.

Bezugsquelle:
SYSTEMTECHNIK BIES
Am Eichbaumeck 47
6100 Darmstadt
Tel.:(06151) 317060

hyperCACHE 030 unter neuem Vertrieb

Ab sofort wird der hyperCACHE 030 der Firma proVME, Neckargemünd, über die Firma Wacker-Systemelektronik vertrieben. Das Beschleuniger-Board für ST-Computer wird in einer 25-MHz-(1489,-DM) und in einer 33MHz-Version (1798,-DM) angeboten.

Beide Versionen haben einen 16KB großen Daten- und einen 16KB großen Instruktions-Cache mit jeweils 32Bit Breite. Hierdurch erreicht das 25MHz-Modell eine Leistung, die dem ATARI-TT in der 32MHz-Version entspricht. Die 33MHz-Version ist etwa 30% schneller als ein ATARI-TT. Bei beiden Modellen kann wahlweise auch auf eine auf dem Board befindliche 68000er-CPU umgeschaltet werden. Dadurch ist es z.B. auch möglich, die auf dem Markt erhältlichen DOS-Emulatoren weiterhin zu benutzen.

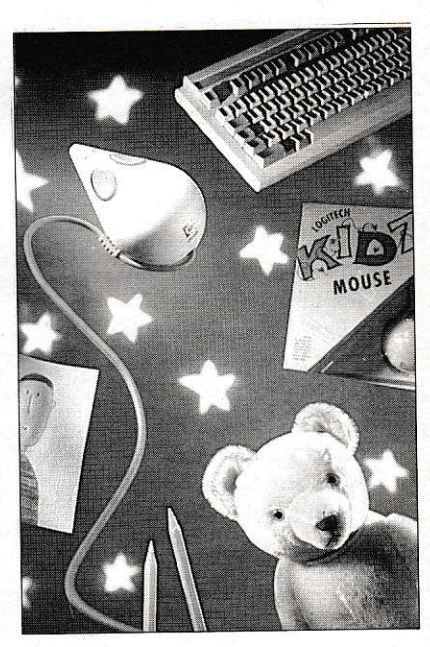
Beide Versionen des hyperCACHE 030 sind ab Lager lieferbar. Ein Einbauservice kann zum Preis von 98,-DM in Anspruch genommen werden.

Bezugsquelle:
Wacker-Systemelektronik GmbH
Bachstraße 39
W-7500 Karlsruhe 21
Tel.:(0721) 551968
Fax: (0721) 593723

KIDZ-Mouse - eine Maus für Kinderhände

Die Schweizer Mäusemacher aus dem Hause Logitech haben sich etwas ganz Besonderes einfallen lassen: Eine Maus speziell für Kinderhände, die in Funktion und Design den Mäusen für Erwachsene in nichts nachsteht. In Verbindung mit speziellen Lernprogrammen soll die KIDZ effektives und leichtes Lernen für Kinder ermöglichen. Sie arbeitet mit 200 dpi Auflösung und ist 100% Microsoft-kompatibel. Eine Version für Atari-ST/STE/TT wird für Anfang 1992 erwartet.

Logi Computer
Tel.:(089) 588071



Neue TOS-Versionen

Das Gerücht kursierte schon seit der ATARI-Messe im August diesen Jahres. Das aus dem Mega-STE bekannte TOS 2.05 soll offiziell für alle „alten" STs freigegeben werden. Nun hat ATARI aber doch einen größeren Schritt getan und zwei neue TOS-Versionen angekündigt. Das TOS 2.06 ist zunächst als Update für die Computer der Mega-STE-Serie gedacht. Laut Informationen von ATARI haben sich unter anderem folgende Änderungen ergeben: Beim Booten wird das Atari-Logo auf dem Bildschirm dargestellt und ein Speichertest durchgeführt, der durch Tastendruck abgebrochen werden kann; nach einem Kaltstart wird während der Wartezeit für die Festplatte ein schrumpfender Balken dargestellt, auch dieser Vorgang kann durch Tastendruck abgebrochen werden; wird während des Boot-Vorganges die CTRL-Taste festgehalten, werden AUTO-Ordner-Programme Accessories und die Datei NEWDESK.INF sowie DESKICON.RSC nicht geladen; per Ziffernblock in Verbindung mit der ALT-Taste kann ein Zeichen durch Eingabe des ASCII-Codes erzeugt werden; die Media-Change-Erkennung bei Floppies wurde verbessert und die Time-Out-Wartezeit bei leeren Diskettenlaufwerken verkürzt. Das TOS 2.06 kann bei den Atari-Fachhändlern zum üblichen Preis von 198,-DM bezogen werden. Durch einen speziellen Adapter (erhältlich über Drittanbieter) kann das TOS 2.06 auch in alle „alten" STs eingesetzt werden.

Neu ist auch das TOS 3.06 für TTs. Hier sind zusätzlich zu den Änderungen, die auch im TOS 2.06 erfolgt sind, noch einige weitere Überraschungen enthalten. Dieses TOS unterstützt nun endlich den 1.44MB-HD-Betrieb mit Diskettenlaufwerken. ATARI wird dazu demnächst einen HD-Umrüstsatz für alle älteren TTs anbieten, der aus dem neuen TOS 3.06, einem HD-tauglichen Einbaulaufwerk und einem speziellen Floppycontroller für den HD-Betrieb besteht.

Update-Service per DFÜ

Die Firma Sigma-Soft, bekannt durch Produkte wie TurboASS und Midimaze II, bietet seit Oktober auch einen Update-Service ihrer Software-Produkte per Mailbox an. Um diesen Service zu nutzen, sollten sich die registrierten Anwender zunächst in der Mailbox eintragen und anschließend eine Postkarte an Sigma-Soft schicken, die Name, User-Name in der Box, Adresse und Seriennummern der entsprechenden Produkte enthält. Sodann kann man jederzeit die neuesten Versionen der Programme downloaden. Auch Freeware-Versionen der Shareware-Programme können in der Sigma-Soft-Mailbox downgeloaded werden. Die Mailbox hat die Nummer (040) 5267185 und unterstützt die Baud-Raten 300-14400 Baud mit V32bis, V42bis, MNP und HST.

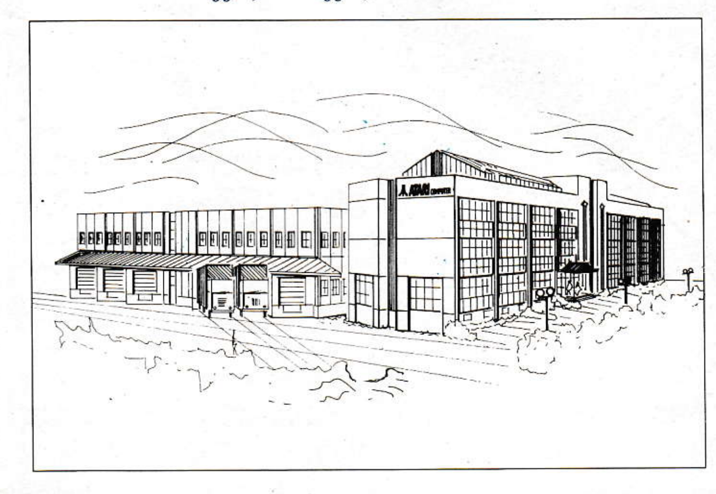
Sigma-Soft
Birkhahnkamp 38
W-2000 Norderstedt 1

Mammut-RAM für Mega-STE

Speichererweiterungen bis auf 16 Megabyte für ATARI-STs sind mittlerweile keine Besonderheit auf dem Markt mehr. Jetzt gibt es auch für die Computer der Mega-STE-Serie eine Speicherkarte, die sich auf bis zu 12 Megabyte ausbauen läßt, wobei der bereits eingebaute Speicher des Mega-STE mitbenutzt wird. Anbieter dieser Karte ist die Firma GENG-Tec aus Mettmann. Die Erweiterung findet im VME-Bus-Slot des Mega-STE Platz und belegt den Adreßraum $400000 bis $BFFFFF, der als Fast-RAM verwaltet wird. Zusätzlich besitzt die Karte einen Steckplatz für das reSOLUTION-Farbgrafiksystem, ebenfalls von GENG-Tec. Die RAM-Erweiterung wird für 8 Megabyte (998,-) bzw. vollausgebaut für 12 Megabyte Gesamtausbau (1398,-DM) angeboten bei:

GENG-Tec
Teichstraße 20
W-4020 Mettmann
Tel.:(02104) 22712

Schaffe, schaffe, Häusle baue....



Die ersten Pfeiler des neuen ATARI-Lager- und Verwaltungsgebäudes in Schwalbach am Taunus stehen bereits; jetzt sind auch die ersten Bilder (noch in Form von Strichzeichnungen) des 20-Mio.-DM-Objektes zu begutachten. Das 4geschossige Gebäude mit rund 6000qm bebauter Fläche ist ein unübersehbarer Beweis für das Vertrauen, das ATARI in den deutschen Markt hat. Die Höhe der Investition unterstreicht die Bedeutung des Standortes Deutschland für Europa, die insbesondere durch die Öffnung Osteuropas noch gestiegen ist. Neben dem Verwaltungsgebäude entsteht auch ein großflächiges Lager, um den künftigen Anforderungen an den Vertrieb mit der entsprechenden Kapazität zu begegnen.

Großzügig ausgestattete Büros, Testräume für Technik und Support gehören ebenso zu dem Gebäudekomplex wie Schulungsräume für Kunden und Anwender. Um der zugespitzten Parkplatzsituation Herr zu werden, ist zudem auch noch eine Tiefgarage mit 90 PKW-Stellplätzen vorgesehen.

Bereits im Spätsommer 1992 soll der Bau abgeschlossen sein und die komplette ATARI-Mannschaft ihre bisherigen Gefilde in Raunheim verlassen können.

Phoenix 1.5

Nachdem Application Systems Heidelberg vor gut einem halben Jahr ihre Datenbank Phoenix vorgestellt hat, ist nun das erste Update auf die Version 1.5 erhältlich. Viele Wünsche und Verbesserungsvorschläge der Anwender sind in diese Version eingeflossen. So ist jetzt in den Manager eine Batch-Verarbeitung integriert, mit deren Hilfe sich häufig wiederholende Prozesse einfach definieren und ausführen lassen. Funktionen wie u.a. Ausführen von Rechnungen, Öffnen der Eingabemaske für eine bestimmte Tabelle, Im- und Export von Datensätzen oder Ausführen von Reports können mittels der Batch-Sprache vollkommen automatisiert werden. Darüber hinaus sind Änderungen in der Report- und Erweiterungen in der Abfragedefinition erfolgt. Der Rechenpuffer wurde vergrößert, um auch komplexere mathematische Berechnungen zu ermöglichen. Die Menüstruktur, Funktionstastenbelegung und das Dialog-Handling haben ebenfalls eine Überarbeitung erfahren. Der Designer wurde u.a. um frei positionierbare Aktionsknöpfe ergänzt.

Das Update-Paket besteht aus einer Diskette, die in komprimierter Form Manager, Designer und ein Installationsprogramm enthält, sowie einer Zusatzdokumentation, in der alle neuen Funktionen beschrieben werden. Registrierte Phoenix-Anwender können das Update auf die Version 1.5 für 70,-DM beziehen. Der Neupreis für das komplette Phoenix-Paket 1.5 beträgt nun 448,-DM.

Bezugsquelle:
Application Systems Heidelberg
Englerstraße 3
W-6900 Heidelberg 1
Tel.:(06221) 300002
Fax: (06221) 300389

Berichtigung

Leider hat auch in der letzten Ausgabe der ST-COMPUTER wieder der Druckfehlerteufel zugeschlagen. Diesmal im Artikel „Fraktale“ (Ausgabe 11/91, Seite 105, Kasten: Komplexe Zahlen).
Bei der Formel zur Division komplexer Zahlen müssen r2 und s2 durch r^2 bzw. s^2 ersetzt werden.
Bei der Beschreibung der Erweiterung der Julia-Mengen auf beliebige Exponenten (Seite 110, erste Spalte, oben) muß die Iterationsformel richtig heißen:

$$z_{k+1} = \sqrt[n]{z_k - c} = (z_k - c)^{1/n}$$

Wir bitten diese Fehler zu entschuldigen.

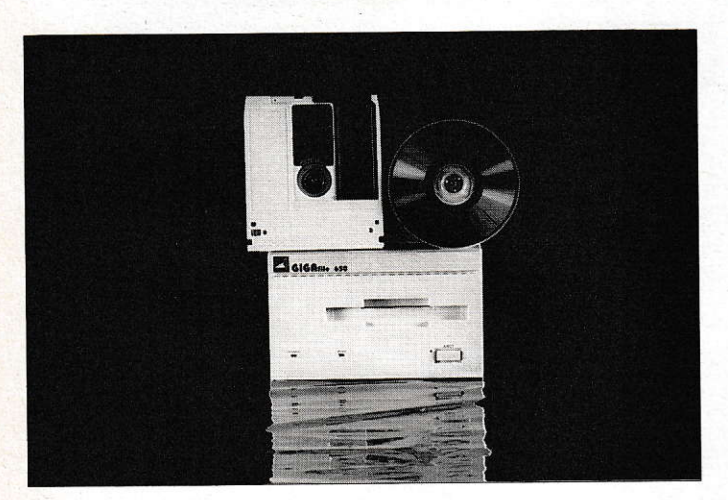
Achtung, XCONTROL!

Aufgrund eines Mißverständnisses mit der Firma ATARI wurde das modulare Kontrollfeld XCONTROL ursprünglich auf die ST-PD-Diskette 467 aufgenommen. Eine Verbreitung als Public Domain ist aber leider aus rechtlichen Gründen **nicht** möglich. Aus diesem Grund wird die ST-PD 467 ohne XCONTROL ausgeliefert. Damit Sie allerdings trotzdem zu einem neuen Kontrollfeld kommen, hat MAXON Computer mit ATARI einen Lizenzvertrag über XCONTROL geschlossen. Es ist somit ab sofort als Sonderdisk 69 (s. Seite 207) erhältlich. Wir weisen aber gleichzeitig darauf hin, daß die Sonderdiskette einem Copyright unterliegt und auf keinen Fall frei weiterkopiert werden darf, also **nicht** Public Domain ist.

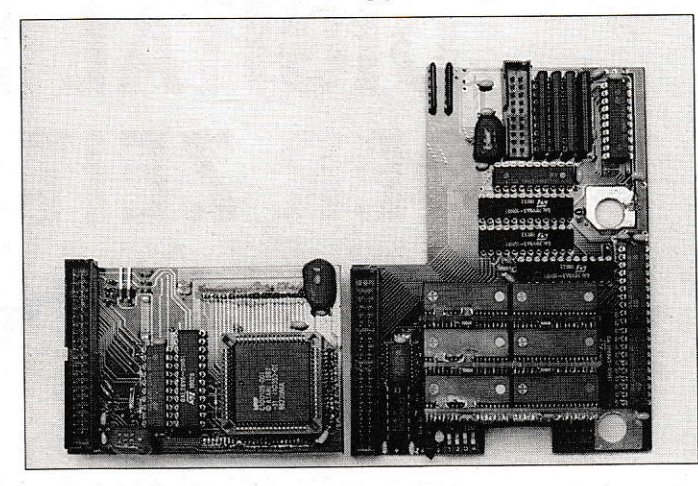
Neue magneto-optische Massenspeicher

Das magneto-optische Wechselplattensystem GIGAFILE 650 wird ab sofort mit neuem Outfit ausgeliefert. Doch nicht nur die äußerlichen Merkmale zeichnen die neue GIGAFILE 650 aus. Folgende technische Neuerungen sind hinzugekommen: TÜV/GS2-Prüfnummer, Handbuch in deutsch/englisch/französisch und ein Notauswurfwerkzeug. Mittlere Zugriffszeiten von unter 60 msek. wurden durch das neue Softwarerelease 2.22 erreicht. Die Datenübertragungsrate liegt bei 680 KB/Sek. Unsere Kunden können für ältere Modelle ein Update anfordern. Der empfohlene Endkundenpreis ist auf 7980,-DM herabgesetzt worden. Die Wechselmedien mit jeweils 2 mal 307 Megabyte (1024 Bytes/Sektor) und 2 mal 281 Megabyte (512 Bytes/Sektor) formatierter Speicherkapazität werden jetzt für 480,-DM angeboten.
Zum Anschluß der GIGAFILE 650 an SCSI-Bus-Systeme, wie beispielsweise ATARI-TT, wird ein Software-Paket zum Preis von 150,-DM angeboten.

Bezugsquelle:
Computersysteme Suplie GmbH
Grenzstraße 158
W-4670 Lünen
Tel.:(02306) 52489
Fax: (02306) 5948



16MB-Erweiterung für Mega-STs



Eine Speichererweiterung auf (fast) bis zu 16MB bietet die Firma Martin Wevelsiep Computertechnik an. Die Karte kann mit speziellen Speichermodulen in 2MB-Schritten aufgerüstet werden. Maximal sind dabei 15.5MByte möglich (etwas geht durch den I/O-Adreßraum des ATARI-ST verloren). Durch die vollsteckbare Bauweise ist der Einbau auch von Laien schnell und sicher durchzuführen, zudem behindert die Speichererweiterung nicht den Einbau einer Mega-Bus-Steckkarte. Auch Beschleuniger-Boards oder MS-DOS-Emulatoren sind weiterhin ohne Probleme einsetzbar. Erhöhte Wärmeentwicklung wird durch den Einsatz der modernen 4MBit-Chips unterbunden. Vorhandener RAM-Speicher kann zudem mitbenutzt und muß nicht ausgebaut oder „totgelegt“ werden.
Der Einbau ist nicht unbedingt auf Rechner der Mega-Serie beschränkt. Auch andere ATARI-Computer, beispielsweise 1040- oder 520-ST, können, soweit es das Platzangebot zuläßt, mit der Speicherkarte ausgerüstet werden.
Die Grunderweiterung, bestehend aus der Platine mit 2MB bestückt, kostet 1198,-DM. Weitere 2MB-Module schlagen mit 429,-DM, 4MB-Module mit 599,-DM zu Buche.

Bezugsquelle:
Martin Wevelsiep
Bogenstraße 32
W-5100 Aachen-Haaren
Tel.:(0241) 167214
Fax: (0241) 165157

Homerecording preisgünstig

Durch die eingebaute MIDI-Schnittstelle ist der ATARI-ST von Beginn an professionell und semiprofessionell zum Musikmachen eingesetzt worden. Um auch angehenden Musikern schon die Möglichkeit zu bieten, mit modernsten Mitteln zu arbeiten, hat ATARI ein speziell auf Hobbymusiker und Einsteiger abgestimmtes Komplettpaket zusammengestellt. Es besteht aus dem bekannten ATARI-1040-STE mit monochromen Monitor SM-124, einem Kawai-Keyboard MS170 samt MIDI-Kabel und dem Sequenzer „Happy Musik“ aus dem Hause Steinberg (bekannt durch den Profi-Sequenzer „Cubase“). Mit dieser Anlage wird jeder Hobby- oder Feierabendmusiker in die Lage versetzt, wie die Profis im Tonstudio zu arbeiten. 12 Spuren können mit dem Sequenzer eingepielt, nachbearbeitet und sogar als Partitur ausgedruckt werden. Das Keyboard bietet 24 Sounds in 16-Bit-PCM-Qualität und zusätzlich 19 Schlagzeugklänge. Daneben können auch 4 eigene Sounds erstellt werden. Mit 1498,- ist das Komplettpaket als äußerst preisgünstig anzusehen und kommt zur Weihnachtszeit sicher gerade richtig.

Neue Grafikkarte von Sang

Nach der Entwicklung der Grafikkarte MEGA-Vision02 vor inzwischen 4 Jahren stellt die Firma Sang nun das Nachfolgeprodukt, die Grafikkarte MEGA-Vision300 vor. Die Karte, die den VME-Slot der Mega-STE- bzw.TT-Computer nutzt, basiert auf dem revolutionären Grafik-Controller INMOS G300, der frei wählbare Punktfrequenzen, Auflösungen und Farbtiefen ermöglicht.
Die darstellbaren Auflösungen liegen zwischen 320x200 und 1280x1024 (noninterlaced). Dabei können 2, 4, 16, 256 (aus 16 Millionen) oder auch 16 Millionen Farben gleichzeitig dargestellt werden. Der 1 Megabyte große Bildspeicher ist mit speziellen Video-RAMs aufgebaut, die extrem schnelles Zeichnen und ergonomische Bildfrequenzen ermöglichen.
Mit einem Endkundenpreis von unter 1500,-DM für die Standardversion bietet die MEGA-Vision300 ein sehr gutes Preis/Leistungsverhältnis.
Die Karte wird für Software-Entwickler ab Januar '92 erhältlich sein. Dem Handel steht die MEGA-Vision300, die auch auf der CeBIT '92 gezeigt wird, ab März 1992 zur Verfügung.

Bezugsquelle:
Sang-Computersysteme GmbH
Kruppstraße 82
W-4300 Essen 1
Tel.:(0201) 820200
Fax: (0201) 8202040

2.5-Zoll-Festplatte jetzt auch extern

Die Aachener Roskothen und Eckstein GbR bietet ihre 2.5-Zoll-Festplatte mit 40MB Kapazität nun in zwei Versionen an. Für alle ATARIs gibt es die EHD-040S mit integriertem Host-Adapter sowie die EHD-040 für Computer mit SCSI-Schnittstelle (z.B. ATARI-TT, Apple Macintosh). Im handlichen Gehäuse mit den Maßen 15x8x5cm (EHD-040S) und 15x8x3cm (EHD-040) läßt sich diese Festplatte in der Hosentasche, im Aktenkoffer sowie im Tresor unterbringen (Datenschutz!). Auch in Universitäten und Gebäuden, in denen Daten von Rechner zu Rechner übertragen werden sollen, ist die EHD-040 ein hervorragender Träger.
Die anschlußfertige Festplatte, bestehend aus der 2.5-Zoll-Platte, Steckernetzteil, Gehäuse, Schalter und LEDs, kostet 1100,-DM (EHD-040S) bzw. 1250,-DM (EHD-040S) und ist ab sofort lieferbar.
Die weiterhin erhältlichen Einbaufestplatten im 2.5-Zoll-Format für Computer der 520- und 1040-ST-Serie haben zu Weihnachten einen kräftigen Preissturz erfahren. Sie kosten jetzt in der 40MB-Ausführung nur noch 1200,-DM (520-ST-Version) bzw. 1250,-DM (1040-ST-Version mit Lüfter). Wer sich scheut, die Winzlinge selbst einzubauen, kann auch den Einbauservice für 70,-DM in Anspruch nehmen.

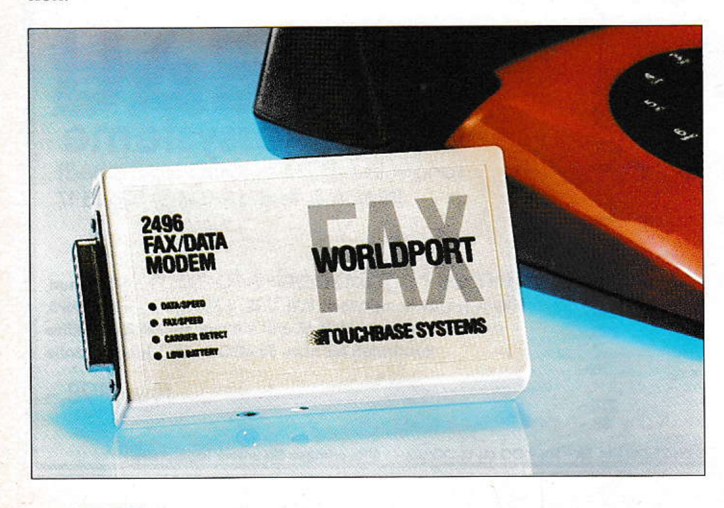
Bezugsquelle:
Roskothen & Eckstein GbR
Monsheimallee 85
W-5100 Aachen
Tel.:(0241) 28840
Fax: (0241) 28842



Pocket-Fax-Modem

Lange erwartet, endlich verfügbar: Ab Oktober ist ein FAX-Modem im Westentaschenformat und mit offizieller ZZF-Zulassung der deutschen Bundespost-Telekom erhältlich. Das Gerät des amerikanischen Herstellers US-Robotics beherrscht neben den üblichen Übertragungsarten V.21, V.22 und V.22bis auch das Fehlerkorrekturverfahren V.42 und Datenkompression nach V.42bis. Dabei sind Geschwindigkeiten von 300 bis 2400 Baud einstellbar. Im FAX-Modus findet die Übertragung mit bis zu 9600 Baud und im Gruppe-3 Standard statt. Durch die geringe Stromaufnahme von maximal 180 mA sind auch längere Übertragungen im Batteriebetrieb möglich.
Als Besonderheit können Verbindungen auch über Akustik-Cups (optionales Zubehör) von jedem Telefon der Welt aus aufgebaut werden. Durch die minimalen Abmessungen ist das „Worldport"-Modem ideal zum Anschluß an den STACY-Laptop, oder an das ST-Book. Alle gängige Software wird durch das erweiterte Hayes-TM-AT-Command-Set unterstützt.
Worldport kostet 1482,-DM und ist erhältlich bei:

MMS Musik Mail Service GmbH
Eiffelstraße 596
W-2000 Hamburg 26
Tel.:(040) 211591
Fax: (040) 211598



Scantool 2.0

Die neueste Version von „Scantool", ein Programm aus dem Hause MARVIN-AG, liegt nunmehr zum Vertrieb aus. Scanner können je nach Komfort die normalerweise vorliegenden Schwarzweißbilder in Grauwerten der Software übergeben. „Scantool" erleichtert nun das Arbeiten mit solchen Grauwertbildern. Selbst schwer zu interpretierende Daten aus Handscannern, mit den üblichen 3*3- oder 6*6-Rastern werden korrekt umgewandelt.
Scantool 2.0 bietet eine direkte Bearbeitung der Gradationskurve per Spline-Funktion sowie Funktionsmakros für Dichte, Kontrast, Mittelton und Gamma-Korrektur, die auch prozentual einstellbar sind. Bilder werden millimetergenau vergrößert und verkleinert bis hin zu druckfähigen Rastern als IMG- oder als Graustufenbilder im TIFF-Format.
Scantool 2.0 kostet 129 SFr, und ein Update der Version 1.0 gibt es für 30 SFr.

Bezugsquelle:
Trillian Computer AG
Eisfeldstraße 6
CH-8050 Zürich
Telefon: 0041/1/302 21 79

NEWS

Mega-Systembus zum Nachrüsten

Ein großer Nachteil der Atari-ST-Serie ist das Fehlen des Systembusses (Mega-Bus), der in den Mega-ST-Geräten implementiert ist. Aus der Hardware-Schmiede des Helmut Schilling stammt das „Mega-Bus-Interface". Diese Erweiterung, die ursprünglich bereits auf der ATARI-Messe vorgestellt werden sollte, erlaubt es, jeden ST-Rechner nachträglich mit dem Mega-Bus auszurüsten. Das Interface wurde so konzipiert, daß es eine „Half-Card" (137x145mm) oder eine „Full-Card" (137x-280mm) aufnehmen kann. Somit ist es erstmals möglich, Grafikkarten (für Großbildschirme etc.) an den kleinen ST-Rechnern der 260-,520- und 1040-ST-Serie zu betreiben. Der Preis der Erweiterung stand bei Redaktionsschluß noch nicht fest.

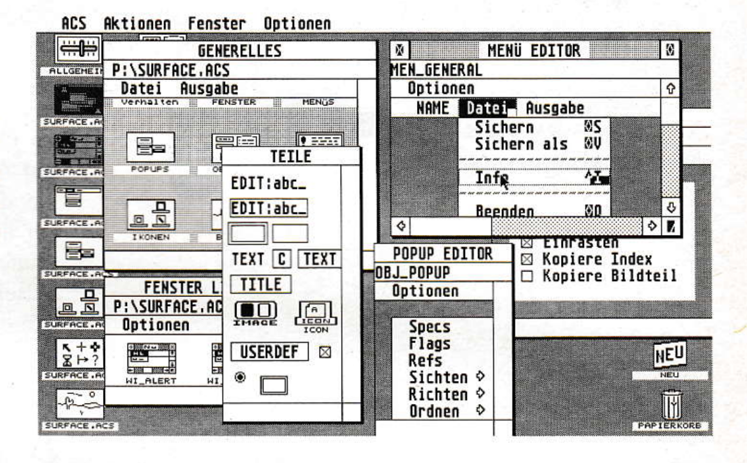
Bezugsquelle:
Helmut Schilling
Willbecker Str.67
W-4006 Erkrath
Tel.:(02104) 449623

Steuer-Software für 1991

Die neueste Version des SteuerStar, der SteuerStar '91, wird ab dem 15.11.1991 ausgeliefert. Die umfangreichen Gesetzesänderungen der großen Steuerreform 1990/91 sind nunmehr im Programm eingearbeitet. Die Möglichkeiten des SteuerStars '91 umfassen unter anderem Lohn- und Einkommensteuerberechnungen 1986 bis 1991, Einkommensteuertabellen 1982 bis 1990/92, Lohnsteuertabellen 1989 bis 1990/92 (Jahres-, Monats-, Wochen- und Tagestabellen A und B) sowie ein Programm zur Berechnung der Einkünfte aus Vermietung und Verpachtung. Der Preis des Programmes beträgt 50,-DM. Updates für registrierte Anwender sind schon für 30,-DM zu bekommen.

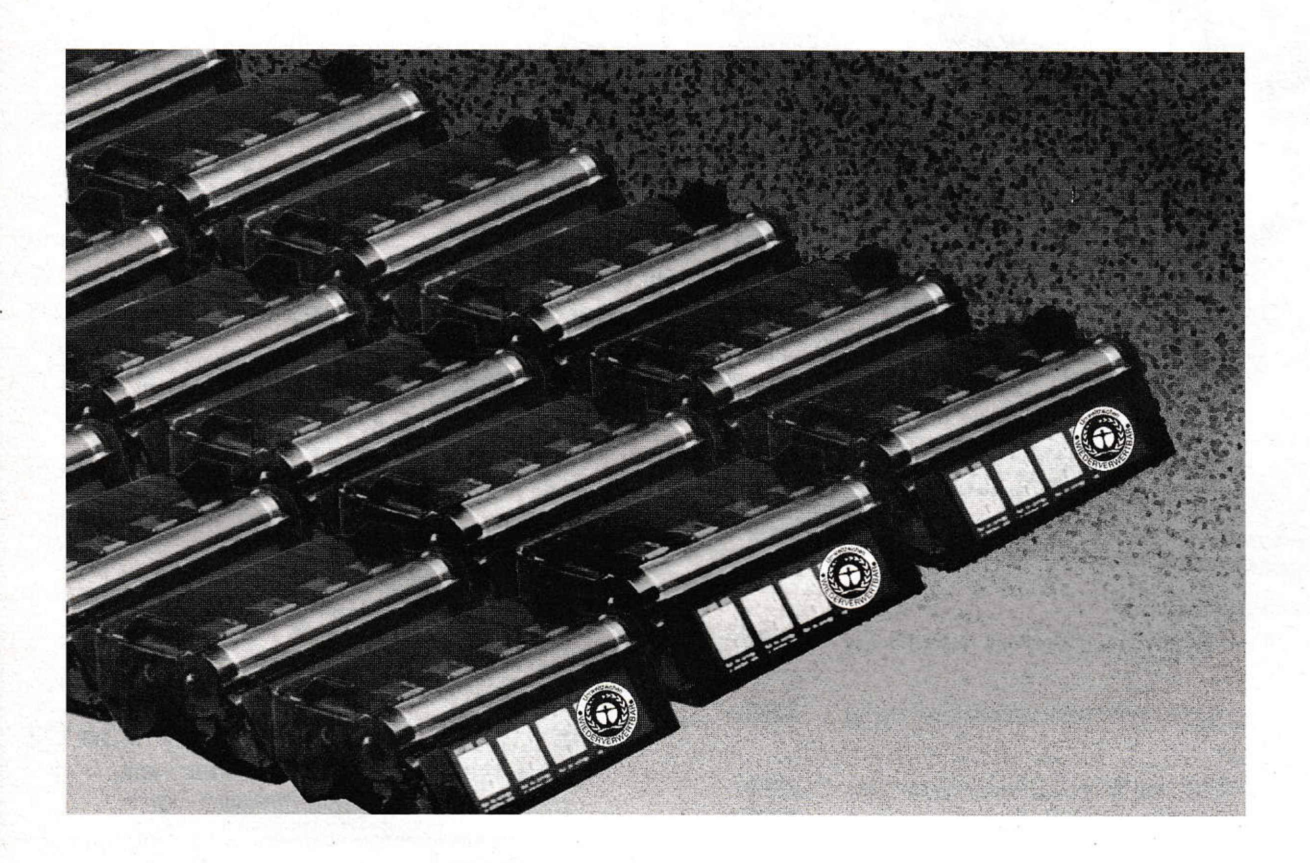
Bezugsquelle:
Dipl. Finanzwirt Jochen Höfer
Grunewald 2a
W-5272 Wipperfürth
Tel.:(02192) 3368

ACS - Ein neues Programmier-Tool



Mit dem neuen Application Construction System, kurz ACS, lassen sich vollständige GEM-Programme in kürzester Zeit erstellen. Es übernimmt dabei lästige Routineprogrammieraufgaben wie z.B. Neuzeichnen von Fenstern, Verschiebeoperationen etc. Mit dem ACS kann sich der Anwender voll und ganz auf seine Anwendung konzentrieren. Er muß lediglich definieren, welche Routinen bei Anwählen von Buttons, Icons etc. auszuführen sind. Das ACS besteht aus einem komfortablen Editor, der u.a. den Funktionsumfang eines RCS und eines Icon- und Image-Editors enthält. RCS-Dateien können weiterverarbeitet werden. Derzeit unterstützt ACS Pure C und Turbo C. Weitere Sprachen wie z.B. MAXON Pascal sind in Vorbereitung. ACS läuft auf ST, STE und TT und wird ab Anfang Dezember für DM 198,- erhältlich sein.

MAXON Computer
Schwalbacher Str. 52
W-6236 Eschborn
Tel. (06196) 481811
Fax (06196) 41885



Toner Recycling

Erst sehr spät hat die Menschheit erkannt, daß das Zeitalter des „Ex & Hopp" nie hätte stattfinden dürfen, und viele beruhigen ihr Gewissen damit, indem sie Altglas und Zeitungspapier gelegentlich zum Container tragen. Eigentlich dürfte die Suche nach wiederverwendbaren Materialien nie aufhören, und jede Idee, selbst die undenkbarsten Gegenstände einer Wiederverwendung zuzuführen, müßte dankbar ergriffen werden.

Sehen wir uns im Haushalt um, fallen uns glücklicherweise viele Gegenstände auf, die in einen Erneuerungskreislauf passen. Im Büro aber wird das schon schwieriger zu überblicken, oder hätten Sie auf Anhieb daran gedacht, nicht ständig neue Toner-Kartuschen für den Laser oder Kopierer zu kaufen?

Bei einem durchschnittlichen Jahresbedarf von vier Toner-Kassetten pro Laserdrucker wandern jährlich fast 3 Millionen Kartuschen auf den Müllberg. Und wenige Zeitgenossen wissen, daß die Toner-Flüssigkeit bzw. das Pulver giftig sind, und alle Behältnisse (besonders auch der Rest-Toner-Behälter) längst zum Sondermüll gehören. Im Grunde ist der ständige Neukauf von Toner höchst unwirtschaftlich und umweltgefährdend zugleich.

Die Firma FARBAX GmbH im südbadischen Albbruck hat sich darauf spezialisiert, in einem ausgeklügelten Recycling-Verfahren die Verbrauchseinheiten für Laserdrucker, Fotokopierer und einige Telefaxgeräte wieder aufzufrischen.

Und so funktioniert das Recycling: Zunächst werden die, vom Anwender eingesandten Kartuschen gekennzeichnet, danach vorgereinigt (evtl. von Rest-Toner befreit) und auf noch vorhandene Funktionsfähigkeit überprüft. Defekte Einheiten werden ausgesondert und speziell weiterverarbeitet.

Nach dieser Vorbereitung heißt es, jede einzelne Kartusche in alle ihre Einzelteile zu zerlegen. Jedes Einzelteil durchläuft eine Sicht- und Funktionskontrolle und wird, soweit nötig, durch neue Teile ersetzt. Defekte Kunststoffteile lassen sich als Granulat wiederverwenden, Metallteile werden eingeschmolzen.

Jetzt beginnt der eigentliche Nachfüllprozeß. Das Unternehmen garantiert die Verwendung von hochwertigem, absolut einwandfreiem Toner. Nachdem die Kassette aufgefüllt ist, wird sie fachmännisch zusammengesetzt und erst nach fünf bis zehn Probeausdrucken in Folie luftdicht verschweißt. Nach fünf Tagen erhält der Kunde seine ursprüngliche Kartusche zurück. Auf alle von der Firma FARBAX aufgefüllten

Toner-Kassetten und auf die Wartungsteile gibt es ein Jahr Garantie, die eine Gratisreparatur bei defekten Teilen einschließt.

Seit kurzem darf die Firma FARBAX allen regenerierten Toner-Behältern den blauen Umweltengel aufdrücken mit dem Prädikat: „weil wiederbefüllbar". Das macht es auch für den Kunden deutlich, daß der Weg dieser Kartusche in Zukunft wieder in das Recycling-Unternehmen führen muß.

Um den gesetzlich geschützten Umweltengel führen zu dürfen, hat sich die Firma FARBAX zu folgenden Auflagen vertraglich verpflichtet:

1. Jede wiederverwendbare Einheit (Toner-Behälter, Entwickler, Trommel) ist für die Zukunft mindestens fünfmal wiederbefüllbar.
2. Die Kapazität der wiederbefüllten Toner-Kassetten beträgt mindestens 3000 Blatt.
3. Es werden keine Einzelteile aus dem hochgiftigen Cadmium verwendet.
4. Die wiederbefüllbaren Einheiten müssen als solche deutlich gekennzeichnet sein.
5. Toner-Rückstände werden, staubdicht verschlossen, einer Müllverbrennung zugeführt.
6. Wiederbefüllte oder regenerierte Produkte müssen mindestens den Qualitätsanforderungen der Originalprodukte entsprechen.
7. Leere, verbrauchte Produkte werden vom Verbraucher zurückgenommen.

Für weit mehr als 500 Laserdruckermodelle, sowie einige Tintenstrahl-, Telefax- und Fotokopiergeräte bietet FARBAX das Nachfüllen von Toner-Kassetten und Wartungsteilen an. Über 4000 Zubehörteile werden monatlich neu aufbereitet. Schweizer Gewissenhaftigkeit und Präzision (schließlich ist FARBAX ein Tochterunternehmen des Schweizer Pioniers im Recycling), Know-How und Qualität zählen mit zur umweltbewußten Firmenphilosophie.

Aber nicht nur Toner-Auffüllen (Toner Refill), auch ein sogenanntes Trommelauffrischen (OCP-Drum Kit Refresh) und der Ersatz von Wartungs-Kits stehen mit auf dem Programm. Gerade wenn der Laserdrucker bei Dauerbelastung sehr schnell blassere Blätter hervorbringt, ist die Haftungsqualität der hochpolierten Trommel nicht mehr sehr hoch. Dann würde man für teueres Geld eine neue Trommel erwerben, ohne zu wissen, daß die auch wiederverwendet werden kann.

Der Atari-Laserdrucker SLM 804 entspricht dem Laufwerkstyp Kyocera F-1010, der SLM 605 ist baugleich mit dem TEC 1305. Als Anhaltspunkte hier die aktuellen Preise:

SLM 804 Toner recyclet = DM 25
Trommeleinheit aufgefrischt = DM 209

SLM 605 Toner recyclet = DM 28
Trommeleinheit aufgefrischt = DM 175

Je nach Modell können die Kosten für Verbrauchsmaterial damit erheblich gesenkt werden, und somit dürfte nicht nur der Umweltgesichtspunkt zur Änderung unserer Wegwerfgewohnheiten führen.

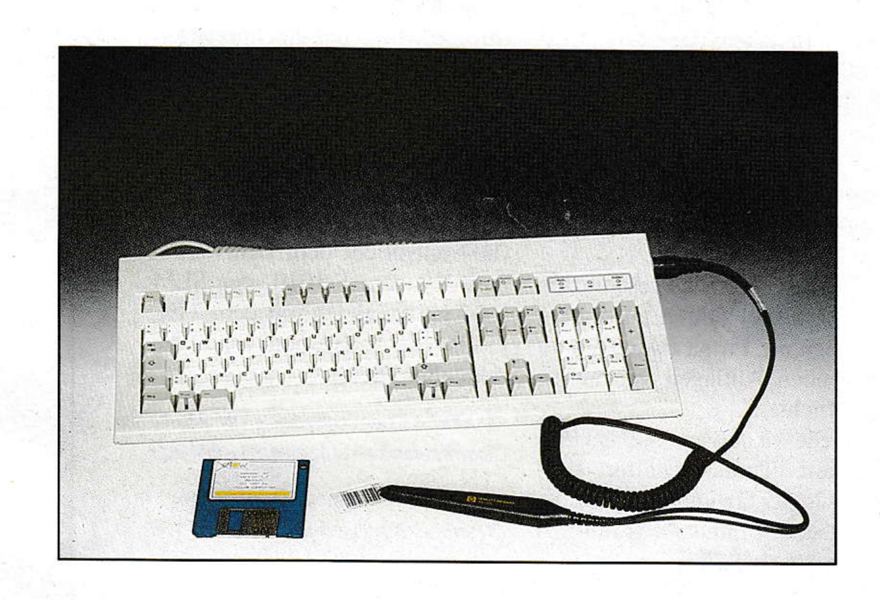
Fordern Sie ausführliches Informationsmaterial, eine Kompatibilitätsliste mit Preisinformationen an. Der Neukunde erhält einen Gutschein für den ersten Auftrag über 20%. Bei einer größeren Abnahmemenge ab 4 bzw. 12 Einheiten gibt es noch einmal einen Preisnachlaß.

Bezugsquelle:

FARBAX GmbH
Alemannenstraße 3
W-7892 Albbruck-Buch
Telefon: 07753 / 2078

Punkt, Punkt, Komma, Strich

Seit etlichen Jahren schon begegnen sie uns täglich. Auf vielen Produkten des täglichen Lebens sind sie zu finden. Kaum einer weiß, was sie bedeuten und wozu sie überhaupt da sind. Natürlich ist hier die Rede von den Bar-Codes. Jene mystisch anmutenden Strichmuster, mit seltsamen Zahlencodes unterlegt (nein, es sind leider nicht die Lottozahlen der nächsten Wochenziehung).



Lange Zeit blieb der Daseinszweck dieser Strichmuster im Dunkeln. Mit Computern müsse es etwas zu tun haben. Mittlerweile hat sich das Geheimnis etwas gelüftet. Seit es in vielen großen Supermärkten sogenannte Scanner-Kassen gibt, kann man dort den Einsatz der Bar-Codes live erleben. Per Laserstrahl gelesen werden aus den Strichen Zahlen, zu denen wiederum im Zentralcomputer Artikelbezeichnungen und, ganz wichtig, die aktuellen Preise gespeichert sind. Lästiges und zeitraubendes Eintippen an den Kassen entfällt. Dies setzt natürlich voraus, daß auch wirklich alle Artikel mit einem Bar-Code bedruckt sind.



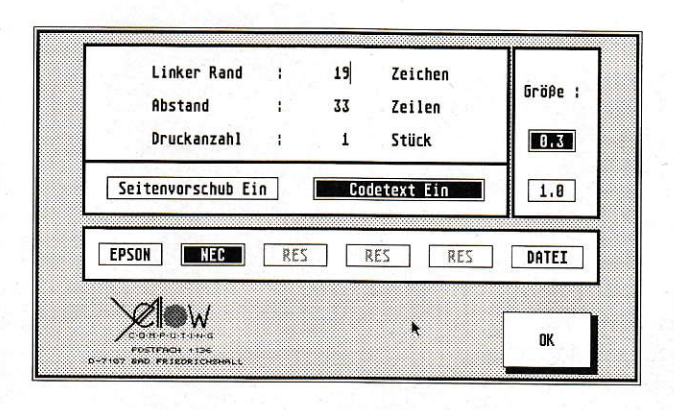
Wie so oft, findet auch hier eine Entwicklung vom High-Tech-Einsatz im professionellen Bereich hin zu Anwendungen in Kleinbetrieben oder sogar zur Privatnutzung statt. Durch die neue Tastatur mit integriertem Bar-Code-Lesestiftanschluß aus dem Hause Cherry, made in Germany, kann man diese fortschrittliche Technologie der Datenerfassung jetzt auch in kleinerem Rahmen sinnvoll einsetzen. Die Tastatur baut auf dem wohl bekanntesten Cherry-Modell G80-1000 auf. Diese bewährte Tastatur wurde äußerlich nur um zwei Sondertasten und eine 6polige Anschlußbuchse an der rechten Seite erweitert. Hier findet der mitgelieferte Bar-Code-Lesestift Anschluß. Auch an ATARI-ST-Computern kann diese Tastatur angeschlossen werden. Hierzu bietet sich das schon einmal in einer früheren Ausgabe der ST-COMPUTER (7/8 1991) vorgestellte Interface der Firma HG-Computer an. Damit ist der Betrieb der Tastatur samt Bar-Code-Leser sofort möglich. Aber auch Tastatur-Interfaces anderer Hersteller lassen sich verwenden, sofern sie den Anschluß einer PC/AT-kompatiblen Tastatur unterstützen.

Universal einsetzbar

Der große Vorteil der Cherry-Tastatur ist, daß zum Betrieb des Bar-Code-Lesers keine spezielle Software benötigt wird. Nachdem der Bar-Code mittels des Lesestiftes eingelesen wurde, sendet die Tastatur ohne weiteres Zutun des Anwenders die entschlüsselten Daten als Tastencodes an den Computer. Dieser braucht also gar nicht zu wissen, ob die Zeichen von Hand eingegeben wurden oder vom Lesestift kamen. Dadurch wird das Gerät für alle Programme einsetzbar, die eine Dateneingabe über Tastatur ermöglichen (welches Programm kann dies nicht?). Selbst die Eingabe von „Return" nach dem Bar-Code kann man sich ersparen, dazu lassen sich über die Tastatur ein Header- und ein Terminator-String programmieren, die bei jedem erkannten Bar-Code vor bzw. nach diesem gesendet werden. Diese Strings können 12 beliebige Zeichen der Tastatur enthalten, somit auch die Return-Taste. Anwendungen wären zum Beispiel eine Lagerverwaltung oder eine Scanner-Kasse für den Kleinbetrieb.

Bar-Codes drucken

Mit der Tastatur und dem Bar-Code-Lesestift sind also schon diverse Anwendun-



BAR-CODE-ST bietet eine einfache Bedienung durch eine übersichtliche Dialogbox.

gen machbar. Allerdings beschränken sie sich auf Produkte, die schon mit den entsprechenden Bar-Codes ausgestattet sind. Will man eigene Codes erstellen und ausdrucken, benötigt man eine spezielle Software. Auf diesem Gebiet ist seit einiger Zeit ein Produkt aus dem Hause Yellow-Computing auf dem Markt. Mit „Bar-Code-ST" lassen sich beliebige Bar-Codes nach EAN13, EAN8 und CODE39 erstellen und ausdrucken. Codes nach EAN8 bestehen aus insgesamt 7 Ziffern plus einer Prüfziffer, die eine Art Check-Summe der vorangegangen Ziffern darstellt. Ähnlich sehen die Codes nach EAN13 aus, hier sind es aber 12 Ziffern plus einer Prüfziffer. Letztere sind die gebräuchlichsten Bar-Codes, die auf fast allen Produkten im Handel zu finden sind. Positiv ist, daß das Programm die Prüfziffer selbst generiert bzw. korrigiert, falls sie nicht korrekt eingegeben wurde. Hierdurch ist immer gewährleistet, daß der Bar-Code im korrekten Format ausgegeben wird. Zusätzlich zu den reinen Ziffern-Codes kann das Programm aber auch Buchstaben in Bar-Codes verwandeln. Hierzu steht der sogenannte CODE39 zur Verfügung. Damit lassen sich Großbuchstaben von A bis Z, Zahlen und Zeichen wie Plus und Minus in Bar-Codes ausdrücken. Eine Zeichenkette kann dabei maximal 18 Zeichen enthalten. Eine nützliche Sache, wenn man Artikel mit ihren Namen oder Kurzbezeichnungen verschlüsselnwill.

Das Programm bietet die Möglichkeit, die eingegebenen Daten direkt als Bar-Code zu drucken oder als GEM-IMG-Grafik auf Disk oder Festplatte zu speichern. Mitgeliefert werden Druckertreiber für EPSON- und NEC-kompatible Drucker. Erweiterungen sind aber vorgesehen. Ein Laser-Treiber soll folgen. Für die Massenerstellung von Bar-Codes haben sich die Software-Entwickler von Yellow-Computing etwas Besonderes einfallen lassen. Über eine einfache Batch-Programmierung läßt sich eine beliebige Anzahl von Bar-Codes erstellen und ausgeben. Dazu braucht lediglich eine Textdatei mit den entsprechenden Anweisungen erstellt und mit der Batch-Option des Programmes ausgeführt zu werden. Praktisch ist dabei, daß sich alle Funktionen, die das Programm bei der manuellen Bedienung bietet, auch über die Batch-Programmierung nutzen lassen. Selbst zukünftige Erweiterungen werden dabei berücksichtigt.

Zusammenfassung

Die Cherry-Tastatur G80-1600 samt Lesestift und das Programm „BAR-CODE-ST" der Firma Yellow-Computing bilden ein gutes Gespann. Bar-Codes, die mit Bar-Code-ST ausgedruckt wurden, lassen sich einwandfrei mit dem Lesestift erkennen. Selbst auf einfachen 9-Nadeldruckern ausgedruckte Bar-Codes werden noch problemlos erkannt. Die Software erfüllt ihren Zweck und ist durch die Batch-Möglichkeit auch in der Lage, Massenverarbeitung durchzuführen. Leider ist sie nicht großbildschirmfähig, und auch auf dem ATARI-TT versagt sie ihren Dienst. Im allgemeinen Zuge der Entwicklung dürfte aber eine diesbezügliche Anpassung nicht lange auf sich warten lassen. Für alle, die professionell mit Bar-Codes arbeiten müssen oder dies zukünftig wollen, sind sowohl die Tastatur als auch die Bar-Code-Software eine lohnende Anschaffung.

CM

Bezugsquellen:

Tastatur/Lesestift/Interface:
HG-Computersyteme
Krugenofen 88-90
W-5100 Aachen
Tel.:(0241) 603252

Bar-Code-ST-Software:
Yellow-Computing
Friedrichshaller Straße 66
W-7107 Bad Friedrichshall
Tel.:(07136) 4097

BARCODE - ERMITTLUNG V 1.0
Autor : Michael Hartmann
EAN8 | EAN13 | CODE39 | RES | RES
BATCH
PARAMETER
Codenummer : ST-COMPUTER______
ABBRUCH
START
ST-COMPUTER
yellow COMPUTING
POSTFACH 1136
D-7107 BAD FRIEDRICHSHALL

In der Parameter-Box lassen sich alle wichtigen Einstellungen tätigen.



MIDI-FUTURE

Es war einmal, in grauer Vorzeit, ein Organist, der jeden Abend emsig seine Orgel programmierte. Alle Regler- und Knöpfchenstellungen wurden genauestens notiert, damit der Sound auf der Bühne auch noch genauso schön war wie daheim. In der Tanzpause war alles flugs umprogrammiert, so daß es auch schnell weitergehen konnte. Dies reichte dem Organisten nicht mehr, und es war ihm auch viel zu stressig, immer alles neu einzustellen. Das ging auch einfacher. Er stellte seinen SUPER-SOUND ein und klebte kurzerhand alle Regler mit Gaffa-Tape fest. Dann zog er los und kaufte eine neue Orgel. Von nun an war das Leben für ihn viel leichter geworden. Zwei Sounds hatte er jetzt jederzeit zur Verfügung. Ach, wenn man doch auf Knopfdruck die gesamten Keyboard-Einstellungen ändern könnte... und auch dies wurde möglich.

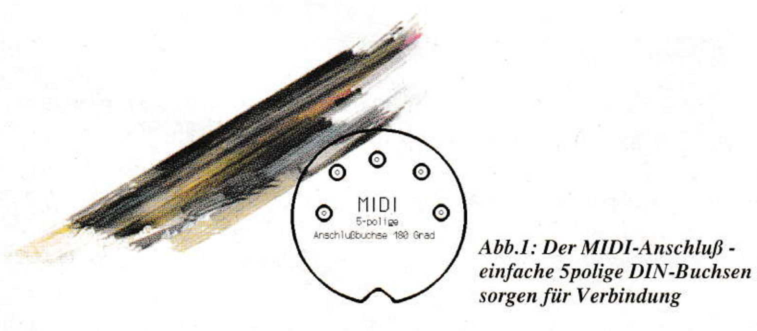


Abb.1: Der MIDI-Anschluß - einfache 5polige DIN-Buchsen sorgen für Verbindung

Es begann eine Zeit, in der die Hersteller gute Orgel- nebst Synthesizer-Sounds in ein einziges Gerät zu integrieren begannen. Ein RAM-Speicher erlaubte das komfortable Abspeichern der selbst kreierten Sounds. Die Rettung aller Bühnenmusiker. Die nun folgende Sound-Flut brachte wieder neue Wünsche zutage. Die Keyboards bekamen Cartridge-Slots, in die dann kleine externe Erweiterungsspeicher eingesteckt werden konnten. Als Massenspeicher war dies nicht der Weisheit letzter Schluß. Speicher kostet nun mal eine Menge Geld. Man dachte da schnell an Disketten, die im Gegensatz zu einer RAM-Cartridge nur einen Bruchteil kosten. Was lag also näher, als einen Computer zur Archivierung für seine Sounds einzusetzen? Die Programmierer ließen nicht lange auf sich warten und stellten sehr bald für alle nun auf den Markt kommenden Keyboards Soundmanager und Bankloader zur Verfügung. Heute wäre es kaum noch denkbar, sich ein Keyboard zu kaufen und kein dazugehöriges Editor-Programm. Den eigentlichen Datenaustausch zwischen Keyboard und Computer ermöglicht die genormte MIDI-Schnittstelle. MIDI ist die Abkürzung für MUSICAL-INSTRUMENT-DIGITAL-INTERFACE.

Atari-ST/TT-Computer besitzen diese Schnittstellen bereits ab Werk, was auch heute immer noch einzigartig ist. Wer sich ein MIDI-Keyboard zulegt, kann sofort mit einem entsprechenden Programm und seinem ATARI loslegen. MIDI-Kabel nicht vergessen. Wer seinen ATARI mal von der Seite betrachtet hat, weiß, daß sich auf der linken Seite zwei genormte 5polige 180Grad-DIN-Buchsen befinden. Genau die, die auch bei Kassettenrekordern Anwendung finden. Einmal MIDI-IN und einmal MIDI-OUT. Wer nun meint, daß wie beim guten alten Kassettenrekorder Tonfrequenzen übertragen werden, irrt. Hier werden ausnahmslos nur digitale Daten übertragen. Jeder Parameter, der an einem MIDI-Instrument verändert werden kann, muß zunächst digitalisiert werden. Anschließend nehmen die „Nullen" und „Einsen" ihren Weg über das MIDI-Kabel. Nun wird auch klar, warum alles nach strenger Norm ablaufen muß. Ohne Norm - auweh, das Chaos wäre groß.

Bald begnügte man sich nicht mehr mit dem Archivieren von Sounds. Der Wunsch nach Steuerungsmöglichkeiten wurde immer größer. Es folgten die Sequenzerprogramme, Notationsprogramme mit Notendruck, Samplemanager, Lernprogramme, Gehörbildung, Begleitautomaten und, und, und, um hier nur einige zu erwähnen. Die Entwicklung ging sehr rasant vor sich und läßt noch kein Ende absehen. Das wollen wir auch hoffen. Die Datenmengen, die über das MIDI-Kabel geschickt werden, sind teilweise so groß, daß die Übertragungsrate von ca. 31 KBaud nicht mehr ausreichend erscheint. Welche Probleme auftreten können, wissen die Anwender von Sequenzer-Programmen sicherlich am besten. Hier deklariert man eines seiner MIDI-Geräte als Master. Auf diese Weise richten sich die anderen Instrumente, sofern als Slave eingestellt, immer nach dem Timecode des Masters. Einen solchen Timecode nennt man unter anderem SMPTE (sprich: sämpti, SOCIETY OF MOTION PICTURE AND TELEVISION ENGINEERS). Diese Synchronisation diente ursprünglich zur Steuerung von Videoaufzeichnungen. Das exakte Ansteuern jedes Bildes wurde ermöglicht, indem ein Timecode auf einer separaten Spur parallel zu den Bild- und Tonereignissen mitgeschrieben wurde. Die grobe Unterteilung erfolgte in Sekunden und die Feinabstimmung in Frames. Da diese Synchronisationsmöglichkeiten die Zusammenarbeit von Computern, Synthesizern, Keyboards und Bandmaschinen erheblich vereinfachen und nichts mehr dem Zufall überlassen werden muß, hat man sich 1972 entschlossen, den SMPTE-Code als Norm einzuführen.

Die Datenmengen, die eine MIDI-Schnittstelle verarbeiten muß, ist in vielen Fällen so groß, daß die Baud-Rate (31.25KHz) nicht mehr ausreichend ist. Also höchste Zeit, sich nach schnelleren Übertragungsmöglichkeiten umzusehen. Der ATARI stellt sicherlich kein Problem dar, da die CPU (Motorola 68000 bzw. beim ATARI-TT 68030) für MIDI-Anwendungen allemal genug Reserven besitzt, um noch ein paar Dinge nebenbei verrichten zu können.

LAN - Local Area Network

Welche Auswirkungen die Geburt der MIDI-Schnittstelle auf das Leben von Musikern und Computer-Anwendern hat, konnten die Urväter damals nicht erahnen, sonst hätte man sich sicherlich für höhere Übertragungsraten entschieden. Daß diese Möglichkeit, Daten auszutauschen, nicht schlecht ist, beweisen die unzähligen Anwendungen und immer größer werdenden Programme. Der Arbeitsspeicher wird zu klein, die Festplatte reicht nicht mehr aus, Keyboards heißen mittlerweile Workstation. Grund genug, sich nach einer Datenübertragung umzusehen, die einerseits alle bisher gebräuchlichen MIDI-Anwendungen zuläßt und anderer-seits erheblich schnellere Datenströme ermöglicht. In letzter Zeit fällt da immer häufiger das Stichwort „LAN". „Local Area Network" ist eine Schnittstelle, die digitale Daten 30 bis 3000(!) mal schneller überträgt als MIDI. Die Beschränkung auf 16 MIDI-Kanäle entfällt hier, da unzählige Zuordnungsmöglichkeiten zur Verfügung gestellt werden. Das MEDIALINK-LAN erlaubt die Adressierung von 253 Geräten, die nach verschiedenen Topologien vernetzt werden können. Je nach Anwendung entscheidet man sich für ein T- oder sternförmiges Netzwerk. Jedes angeschlossene Gerät nennt sich in der LAN-Welt „Node". Alle Nodes werden mit einem Glasfaserkabel verbunden und vermögen alle gleichzeitig zu senden und zu empfangen. MIDI-Geräte können über einen sogenannten „TAP" ohne weiteres in das LAN-Netz eingespeist werden. In dem TAP findet eine Konvertierung des Datenformates sowie eine Umsetzung der Signalformen zwischen elektrischen und Lichtwellenleitern statt. Durch diese Konvertierung erreichen LAN-Dateninformationen ohne Probleme die angeschlossenen MIDI-Geräte. MIDI-TAPs arbeiten mit einer Übertragungsrate von 2 Millionen Bits pro Sekunde (2MBaud). Diese Geschwindigkeit erlaubt es, bis zu 8 komplette MIDI-Stränge gleichzeitig zu verarbeiten. Wird ein MIDI-Strang nicht voll genutzt, sind selbstverständlich weitere Funktion möglich. Die bestehenden MIDI-Kanalzuordnungen werden in keiner Weise beeinflußt, sondern mit für das LAN-Netz notwendigen Adressierungen ergänzt.

Die MIDI-Zukunft sieht somit abwechslungsreicher aus, als wir gedacht haben. Computer werden sicherlich bald mit entsprechenden Schnittstellen ausgestattet, um eine effizientere Datenübertragung zu ermöglichen. Prinzipiell ist alles vorhanden und wird auch schon für andere Anwendungen genutzt. Centronics, RS-232, V24, Inhouse, Ethernet, um hier nur einige zu nennen, haben zur Zeit noch (fast) nichts mit MIDI zu tun. Dies kann sich in nächster Zukunft schnell ändern. Wer in einer der letzten „ST-COMPUTER"-Ausgaben aufmerksam die MIDI-News gelesen hat, dem ist sicherlich aufgefallen, daß die RS232(Modem)- Schnittstelle schon für MIDI-Anwendungen genutzt wird. Der ATARI ist nun mal für fast alles zu gebrauchen. Sei es DTP, CAD oder MIDI, die Entwicklung geht immer weiter, so auch bei den MIDI-Programmen, von denen wir hier 2 an der Zahl vorstellen wollen. Gute Sequenzerprogramme gibt es nun wirklich schon genug, da wird in der nahen Zukunft wohl nichts Weltbewegendes mehr geschehen. Der Trend der letzten Zeit geht eindeutig in Richtung „Begleiten" und „Arrangieren" mit dem Computer. Warum sollte der ATARI nicht genauso gut als Begleitautomat genutzt werden können wie ein Rhythmus-Keyboard und dabei vielleicht noch selbständig Arragements erstellen können? Gegenwärtig entwickeln die Hersteller Keyboards, in denen genau dieselben Prozessoren und Speicher stecken wie in einem Computer. Es werden nun mal nicht alle Möglichkeiten zur Verfügung gestellt, aber theoretisch ist es möglich, mit der Synthesizer-Elektronik eine Textverarbeitung durchzuführen. Das wollen wir hier aber nicht. Wir wollen einmal sehen, was uns die mir zur Verfügung gestellten Programme zu bieten haben.



DUETT KOMPLETT

Begleiten und Arrangieren mit dem ATARI

Freestyle und Session Partner sind 2 Programme, die sehr viel gemeinsam haben, aber dennoch völlig verschieden sind. Freestyle ist eher als Begleitautomat/Arrangierer anzusehen und Session Partner als eine Art Groove-Designer mit umfangreichen Zufallsfunktionen. Beide benötigen externe Klangquellen wie Keyboards oder Expander. Die Ausgabe über MIDI ist frei definierbar. Bei Freestyle sollten mindestens sechs auf verschiedenen MIDI-Kanälen empfangende Tonerzeuger vorhanden sein. Session Partner kann mehr ansteuern, auch wenn es nicht immer sinnvoll ist. Wir wollen beide Programme aber nicht miteinander vergleichen, sondern beide unabhängig voneinander beleuchten.

Freestyle - Begleitung erwünscht

Freestyle ist eine ausgewachsene Arrangier-Software, die es faustdick hinter den Ohren hat. Der Firma Sound Pool aus Berlin ist mit diesem Programm sicherlich ein großer Wurf gelungen. Aber wo viel Licht ist, da ist auch viel Schatten. Ob diese Aussage zutrifft, soll der folgende Testbericht zeigen.

In Abbildung 1 sehen wir den eigentlichen Arbeitsplatz, der starke Ähnlichkeit zu Sequenzerprogrammen aufweist. Die grafisch bedienbare Oberfläche erlaubt das schnelle und langsame Vor- und Zurückspulen, wie im richtigen Leben beim Umgang mit einer Tonbandmaschine. Ein integriertes Mini-MIDI-Mischpult erlaubt das „Muten“ einzelner MIDI-Spuren, so daß dieser Kanal einfach stummgeschaltet wird. Des weiteren können Programmwechselbefehle, Lautstärke, Anschlagdynamik und die Oktavlage eingestellt werden, um jederzeit eine Anpassung an das vorhandene MIDI-Equipment vornehmen zu können. Direkt unter der Menüleiste erhält der Anwender Aufschluß über den freien noch verfügbaren Speicher, Spielmodus (dazu später mehr), das Tempo,

werden nur durch den MIDI-Kanal und die Oktavelage angepaßt. Die Tonhöhe kann um +/-2 Oktaven verändert werden. Zwischenwerte lassen sich hier nicht einstellen.

Die Hauptseite

Nach Programmstart offenbart sich die Hauptseite mit allen Anwahlmöglichkeiten. Von hier aus können alle Funktionen erreicht und der gesamte Songaufbau beeinflußt werden. Auffällig sind die 8 untereinander liegenden Eingabefelder. Hier werden einzelne Parts eingegeben, die dann später hintereinander ablaufen. Etwas weiter oben sehen wir eine der Part-Bezeichnungen. In diesem Fall ist es A1. Dieser Part kann in dieser Zeile durch Löschen (DEL) und Einfügen (INS) verändert werden. Durch Doppelklick auf diese Part-Bezeichnung werden weitere 8 Parts mit Kennung B1-B8 eröffnet. Das Umschalten von Ebene A nach Ebene B und umgekehrt ist nach dem eben beschriebenen Verfahren jederzeit möglich. Hinter Song-Info verbirgt sich ein Notepad, das gleichzeitig mit dem Song abgespeichert werden kann. Das Abspeichern ist übrigens als MIDI-File möglich, um die Weiterverarbeitung durch einen Sequenzer zu gewährleisten.

Des weiteren können Tempo, Stil, Taktart und Tonlage vorgegeben oder aber automatisch generiert werden. Soll alles per Zufallsgenerator erstellt werden, braucht man nur auf die Würfel neben oder unter den entsprechenden Feldern zu klikken. Wie von Geisterhand bewegt, wird ein Instrument nach dem anderen, oder auch nur ein einziges, neu gestylt. Die Akkordwahl eines einzelnen Parts wird bei Klick auf den Part-Würfel zufällig getroffen. Ein Klick auf den Band-Puffer-Würfel stellt alle Instumente neu ein. Jeder Akkord kann durch Anklicken direkt beeinflußt werden. Soll ein Instrument mal ausgeschaltet werden, klickt man auf das Kreuz neben dem entsprechenden Instrumentenwürfel. Dieses sogenannte „Muten" ist sogar möglich, während man sich sein soeben erstelltes Stück anhört und mal probieren möchte, wie es mit oder ohne Gitarre klingt. Man merkt gleich, daß sich die Programmierer große Gedanken gemacht haben, um ein leistungsstarkes Werkzeug bereitzustellen.

Ergebnisse

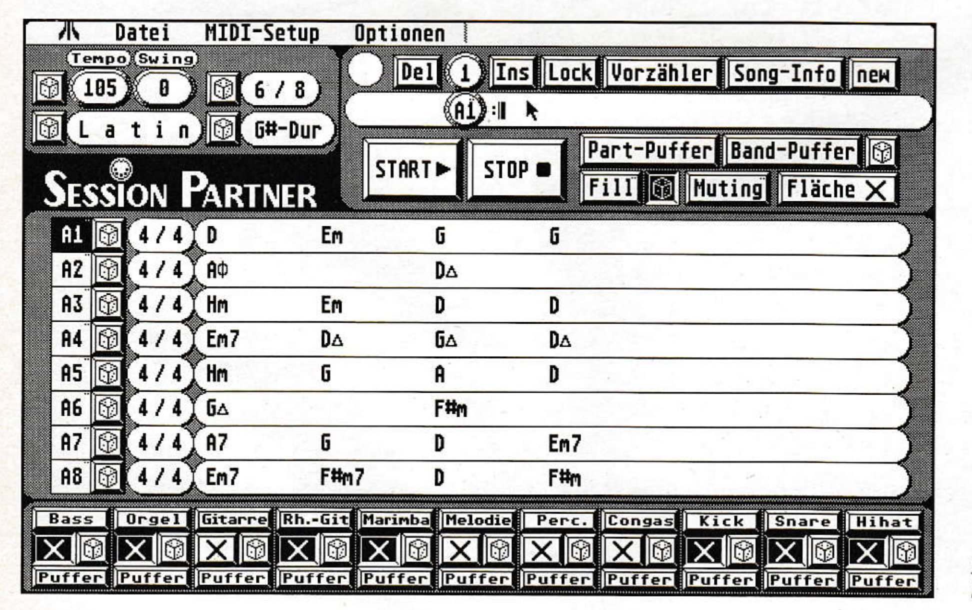
Nach längerem Probieren brachte ich schon recht nette Sachen zustande, die in keiner Weise an ein bekanntes Lied erinnerten, aber in sich doch recht interessant waren. Für nachträgliche Vertonungen kann man sich zur Zeit kein besseres Hilfsmittel wünschen. Eine dominante Melodieführung sollte der Benutzer jedoch nicht erwarten, das kann Session Partner nicht. Die Variationen sind so vielseitig, daß ich bei aller Liebe zu Zahlen nicht in der Lage bin, hier auch nur einen Annäherungswert zu liefern. Die Masse allein macht den Kohl ja auch nicht fett. Es muß schon Geduld aufgebracht werden, um zu einem akzeptablen Ergebnis zu kommen. Letztendlich wird man den letzten Schliff wohl durch einen Sequenzer bewerkstelligen müssen, da die gezielten Operationen dann schnell und reibungslos ablaufen.

Fazit

Um den Diskettenoperationen nicht ein eigenes Kapitel schenken zu müssen, möchte ich hier ein weiteres Lob aussprechen. Diese Option ist umfangreich und nützlich aufgebaut. Session Partner lief während der Testphase nicht immer einwandfei. Sollte der Arbeitsspeicher mal nicht ganz ausreichen, verabschiedet sich das Programm, ohne einem die Chance zu lassen, sein Werk schnell noch zu sichern. Mausbewegungen waren zwar noch möglich, ein Erreichen der Drop-Down-Menüs jedoch nicht mehr. Eine Überarbeitung durch die Programmautoren ist sicherlich notwendig. Dabei sollte man sich dann auf die wirklich wichtigen Funktionen beschränken und einige Nebensächlichkeiten rauswerfen, damit wieder mehr Arbeitsspeicher zur Verfügung steht. Die fehlende MIDI-Thru-Funktion könnte akzeptiert werden, erschwert aber schon beim Einstellen des MIDI-Setups die Kontrolle über die Soundmodule. Hier hätte man weiter denken sollen. Das Timing-Verhalten ist, wie auch in der Bedienungsanleitung erwähnt, etwas problematisch. Der Ursprung liegt dabei aber mehr bei der Datenmenge, die gesendet wird, und nicht bei einer fehlerhaften Programmierung. Mein Expander (U220) hat da bei 3-4 Kanälen schon erheblich mehr zu kämpfen, wenn die Polyphonie bis zum Letzten ausgereizt wird. Da hier 13 MIDI-Kanäle gleichzeitig ihre Informationen hinauspusten, sollte für diesen Fall keine schlechte Beurteilung stattfinden, sondern die vielfältigen Möglichkeiten als Pluspunkt angerechnet werden.

Nun könnte man meinen, bei Session Partner würden die Nachteile überwiegen. Das ist auf keinen Fall so, denn die positiven Aspekte zählen auf jeden Fall und können aus Gründen der Vielzahl hier nicht aufgezählt werden. Der Preis von ca. 200,-DM scheint mir angemessen. Andere Programme, die auf ihrem Gebiet weitaus weniger leisten, kosten in vielen Fällen erheblich mehr. Wer gerne neue Musik erstellen möchte und interessante Varianten komponieren will, hat hier ganz sicher das, was er immer schon gesucht hat, gefunden.

Wolfgang Weniger



Bezugsquellen:

Freestyle:
Fröhlich Musiconsulting
Postfach 1424
W-3550 Marburg 1
Tel.:(06421) 25090

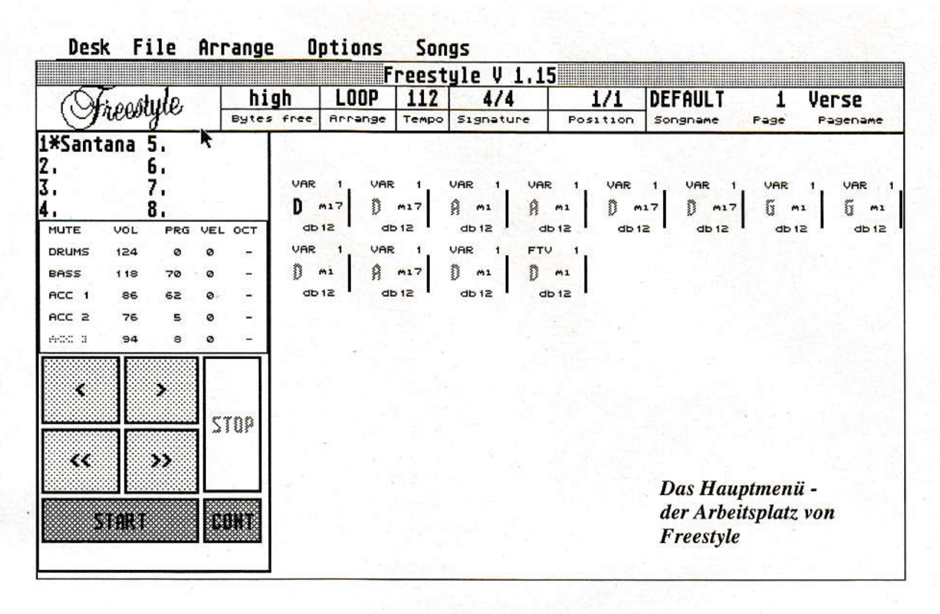
Session-Partner:
DVPI GmbH
Neumühleweg 12
W-7068 Urbach



Die Zentrale des Session Partner

Taktart, Songposition, Songname, Songseite und den dazu gehörigen Seitennamen, wie zum Beispiel Verse, Refrain oder Bridge. Die Namensgebung wird vom Musiker selbst bestimmt oder kann als Default belassen werden.

Freestyle ermöglicht den Zugriff auf 8 verschiedene Grooves, die sich alle gleichzeitig im Speicher befinden und während des Arrangierens beliebig im Song gewechselt werden können. Der in Abb.1 gezeigte Arbeitsplatz ist die sogenannte Leadsheet-Darstellung. Hier sind jederzeit der Songablauf und die aktuelle Songposition durch Fettdruck erkennbar. Freestyle kann als Arrangier-Software oder als Begleitautomat eingesetzt werden.



Das Hauptmenü - der Arbeitsplatz von Freestyle

Menüleiste/File

Mit „load set" werden 10 Songs gleichzeitig in den Speicher geladen, so daß komplette Tanzrunden auf verschiedenen Disketten zusammenkopiert werden können. Mehr als 10 Songs sollten sich dann nicht auf dem Datenträger befinden. Beim Abspeichern sollte darauf geachtet werden, daß kein Unterverzeichnis angelegt, sondern alles unter „ROOT" abgespeichert wird. Da Freestyle auch ohne Monitor eingesetzt werden kann, ist dieses Feature optimal für eine eventuelle Bühnenbenutzung.

Unter „load song" kann ein Song an die erste Stelle der Songliste inklusive aller verwendeten Styles und MIDI-Einstellungen geladen werden, „save song" speichert ihn (wer hätte das gedacht?).

Mit „load style" wird die nächste freie Position der Stylebox mit dem gewünschten Style belegt.

„erase styles": diese nützliche Funktion löscht alle nicht benötigten Styles aus der Stylebox. Dies sollte zum Beispiel vor jedem Abspeichern durchgeführt werden, damit Arbeitsspeicher und Speicherplatz auf dem Datenträger gespart werden können.

Mit der „play list"-Option kann ein Standard-MIDI-File im Format 0 (1 Spur) importiert werden. Dies ermöglicht das Weiterverarbeiten eigener Sequenzerstücke.

„save MIDI file" speichert im MIDI-File-Format 1 (5 Spuren).

„generate Style": hier kann ein Style im MIDI-Format 1 geladen werden. Es kann 13 Tracks enthalten. Da diese Funkion noch viele Möglichkeiten in sich birgt, ist zu empfehlen, die Bedienungsanleitung zur Hand zu nehmen.

Freestyle wird übrigens in einem schönen Etui mit Reißverschluß geliefert. Das ist doch mal etwas anderes. Pappschuber haben wir mittlerweile schon genug herumstehen.

Menüleiste/Arrange

Die unter diesem Menüpunkt versteckten Funktionen gestalten das Arbeiten mit Freestyle erst richtig komfortabel. Entries, so nennen sich hier die Akkordeingaben, können kopiert oder gelöscht werden. Eine direkte Bedienung durch Tastaturkommandos macht das Arrangieren zu einer wahren Freude, da alles sehr schnell vonstatten geht, und nicht immer wieder umständlich ein Menü aufgerufen werden muß. Die Tastaturzuweisungen sind logisch gewählt worden, so daß nach kürzester Zeit die gängigsten Funktionen auswendig beherrscht werden. Alle nötigen Editiermöglichkeiten, wie das Kopieren von Entries, Seiten, Parts und Spieleinstellungen, sind vorhanden, um auf schnellstem Wege zum Erfolg zu kommen. Wer meint, daß sein Song nicht in der richtigen Tonlage komponiert worden ist, kann ihn seitenweise oder aber auch insgesamt beliebig transponieren.

„drumset": diese sinnvolle Einrichtung ermöglicht das komfortable Erstellen einer eigenen Drum-Map, die auch jederzeit auf Diskette abgespeichert und bei Bedarf fest installiert werden kann. Das Problem der Drum-Maps ist allen Sequenzer-Programmierern seit langem ein Dorn im Auge, da nicht alle Expander über separate Editiermöglichkeiten verfügen, um einen Song an ein neues Instrumentarium anpassen zu können. Einige Maps sind bereits vorhanden, so daß einem schnellen Start nichts im Wege steht. Wer sich aber die Zeit nimmt und etwas mit den Notennummern herumexperimentiert, wird feststellen, daß es möglich ist, eine Notennummern mehrfach zu vergeben. Der Kreativität sind somit keine Grenzen gesetzt.

„Program change A+B": die Zuordnung funktioniert prinzipiell genau wie bei den Drum-Maps und kann ebenfalls abgespeichert werden. Es sind Werte von 0 bis 127 möglich. Eine ungültige Eingabe führt zu einer automatischen Korrektur in eine erlaubte Programmwechselnummer.

„MIDI setup": im MIDI-Setup werden Masterkeyboard, Computer und empfangende MIDI-Geräte aufeinander abgestimmt. Die mittlerweile eingebürgerten Grundeinstellungen, die bei vielen Multikeyboards wiederzufinden sind, werden bei Freestyle als Default vorgegeben.

Wer nun schon durch ein entsprechendes Rhythmus-Keyboard vorbelastet ist, bemerkt die Ähnlichkeit zum Roland E-20 und anderen Instrumenten dieser Art schnell. Hier können die schon oben erwähnten 6 MIDI-Kanäle für die Sound-Module eingestellt werden. Wer als Einspiel-Keyboard einen einfachen Synthesizer verwendet, der auf nur einem MIDI-Kanal senden kann, kommt nun in den Genuß, seinen „alten Schinken" splitten zu können. Wenn wir es genau nehmen wollen, ist die Einstellung eines siebten MIDI-Kanals auch noch möglich, nämlich des für das am MIDI-In angeschlossene Gerät. Freestyle kann auf allen „Frequenzen empfangen". Ist der Anwender im Besitz eines Supersounds, der im Baßsowie im Solo- und Flächenbereich gute Dienste verrichten kann, hat er die Freiheit, für Baß und ACC1 denselben Sendekanal einstellen zu können. Jede Begleitstimme kann im Tonbereich durch Eingabe einer tiefsten und höchsten Note eingegrenzt werden. Durch Einstellen des Sync.intern/extern-Parameters wird Freestyle zum „Master" oder zum „Slave" deklariert.

Unter dem Options-Menü sind noch einige weitere sinnvolle Einstellmöglichkeiten vorhanden, wie zum Beispiel „human touch". Wird dieser Parameter aktiviert, werden die Schlagzeuginstrumente

```
File
load set
load song
save song
load style
erase styles
load MIDI playlist
save MIDI file
generate STYLE
quit
```

```
Arrange
add entry        (a)
delete entry     (d)
copy page        (c)
new page         (N)
erase page       (e)
split page       (s)
join page        (j)
copy style       (y)
copy part        (p)
copy mute        (m)
copy octave      (o)
transp. page ⇩   (t)
transp. page ⇧   (T)
transpose song   (S)
arrange list     (l)
```

```
Options
drumset
program change A
program change B
MIDI setup
chord delay       (|)
✓ quick transpose
✓ no MA 6th chord
✓ velocity switch
✓ controller
✓ keyboard split  (k)
human touch
b vs. #
barsplit          (:)
style info        (i)
notepad           (n)
```

Die Menüs von Freestyle im Überblick

Acou BD	36	Cowbell	56
Rim Shot	37	User 4	57
Acou SD	38	User 5	58
Hand Clap	39	User 6	59
Elec SD	40	High Bongo	60
Acou Low Tom	41	Low Bongo	61
Clsd Hi Hat	42	Mt High Conga	62
Acou Low Tom	43	High Conga	63
Open Hi Hat2	44	Low Conga	64
Acou Mid Tom	45	High Timbale	65
Open Hi Hat1	46	Low Timbale	66
Acou Mid Tom	47	High Agogo	67
Acou Hi Tom	48	Low Agogo	68
Crash Cym	49	Cabasa	69
Acou Hi Tom	50	Maracas	70
Ride Cym	51	Samba Whis S	71
User 1	52	Samba Whis L	72
User 2	53	Quijada	73
Tambourine	54	User 7	74
User 3	55	Claves	75

EXIT
SAVE
LOAD

In der DRUM-MAP wird jedem Schlagzeug-instrument eine MIDI-Notennummer zugeordnet

zufällig mal lauter und leiser angesteuert, vergleichbar mit Drumcomputern, die unter anderem auch das Timing in voreinstellbaren Grenzen aus dem Rahmen rükken, um dem Song den sogenannten „human touch“ also kleine menschliche Fehler zu geben.

Prinzip

Fangen wir einfach mal an, einen Song zu kreiren. Freestyle bietet alle Voraussetzungen, ein komplettes Musikstück zu arrangieren. Eine Solostimme kann über den oberhalb des Splitpunktes eingestellten MIDI-Kanal in Realtime dazugespielt werden, ohne diese wie bei einem Sequenzer gleichzeitig mit aufzunehmen.

Machen wir erst einmal alle Seiten mit „erase Page“ frei von Songresten. Ein Entry bleibt übrig. Dieses Entry können wir nun manipulieren, und zwar indem wir mit der linken und der rechten Maustaste die zur Verfügung stehenden Werte durchscrollen. Das betreffende Entry kann als Original (ORG), Variation (VAR), Fill to ORG (FTO), Fill to VAR (FTV), Intro (INT) oder Ending (END) verwendet werden. Sebstverständlich ist es auch möglich, die Tonlage und den Akkordtype zu bestimmen.

Alle gängigen Taktarten sind möglich. Die Auflösung beträgt 1/96tel. Nun deklarieren wir unser erstes Entry als „Intro“, und schon ist der Anfang gemacht. Wie man seinen Song letztendlich aufbaut, bleibt jedem selbst überlassen. Es soll nur prinzipiell aufgezeigt werden, wie das Arrangieren mit den vorgegebenen Hilfsmitteln schnell zum Erfolg führt. Glaubt man, die richtigen Vorkehrungen für einen schönen Song-Anfang gefunden zu haben, wählt man die zweite Seite an, um nun einen neuen Vers, Refrain oder Bridge hinzuzufügen. Langsam aber sicher vervollständigt sich die Songstruktur, und man kann sein Werk entweder als Song mit Anfang und Ende abhören oder aber in einer Endlosschleife (LOOP) laufen lassen. Durch umfangreiche Kopierfunktionen wird die Arbeit sehr beschleunigt. Nun schnell als MIDI-File abspeichern und in ein Squenzerprogramm einladen, um Melodiestimmen und eventuell Effekte zu ergänzen. Fertig ist der Song. Für sich immer wiederholende Passagen können einzelne Seiten verwendet werden, um sie dann komfortabel in beliebiger Reihenfolge aneinanderzuhägen. Dies geschieht in einer eigens für diesen Zweck zur Verfügung gestellten Arranger-List.

„Der kann uns viel erzählen“, werden Sie jetzt bemerken, es geht aber tatsächlich so schnell und einfach. Arrangieren ist damit durchaus ein Kinderspiel. Freestyle kann aber noch mehr.

Der MIDI-Arranger

Wer schon vierstellige Summen locker gemacht hat, um sich ein Rhythmus-Keyboard zu kaufen, möge nun aufmerksam weiterlesen und das Geld lieber gewinnbringend anlegen. Das Arrangieren eines kompletten Songs war ja schon ein recht schnelles Unterfangen, ist jedoch nicht immer das, was man eigentlich im Sinn hatte. Jeder Benutzer will nach den Anpassungen sicherlich einmal hören, wie sein MIDI-Equipment auf die Sendebefehle von Freestyle reagiert. Da liegt es doch nahe, erst einmal eine kleine Session zu veranstalten. Mit „/“ wird der MIDI-Modus eingeschaltet und mit den Funktionstasten gesteuert. Hier tauchen wieder die einzelnen Entry-Versionen auf, die auch schon auf dem Lead-Sheet verwendet wurden. Es ist also alles im Realtime-Betrieb anwendbar und genauso zu handhaben wie ein entsprechendes Keyboard. Durch ein Akkord-Delay wird eingestellt, ob bei einem Akkordwechsel sofort reagiert werden soll oder erst nach dem Taktwechsel. Aus dem Timing kann man also nicht so schnell geraten. Es können jedoch Zwischenwerte eingestellt werden.

Wem ein zweiter ATARI zur Verfügung steht, der hat die Möglichkeit, alles in einem Rutsch auf einer oder (wenn möglich)auf mehreren Spuren gleichzeitig aufzunehmen. Komfortabler geht’s nimma mehr.

Fazit

Eine Sache, die längst nicht bei allen Programmen als Selbstverständlichkeit anzusehen ist, möchte ich nicht unerwähnt lassen. Der Testkandidat lief bei meinen teilweise sehr hektisch ausgeführten Mausoperationen ohne Abstürze und Fehlfunktionen, auch wenn ich durch bewußt falsches Handling Fehler provoziert habe. Ein ganz dickes Lob an die Programmierer. Der Software-Kopierschutz ist aber etwas aus der Mode gekommen, eine Hardware-Lösung per Dongle wäre sicherlich angebrachter.

Als Kritiker mit 2 mal 150 Watt auf der Hutablage, aber nicht mal Licht am Fahrrad, kann ich mir einige negative Punkte nicht verkneifen. Der MIDI-Mixer ist viel zu klein geraten. Die Benutzung eines hochauflösenden Monitors ist somit Bedingung, aber trotzdem noch anstrengend. Die wichtigsten Kopierfunktionen würde ich mir am unteren Bildschirmrand wünschen, damit das Arbeiten auch ohne das Auswendiglernen von Tastaturkommandos zu vollbringen ist. Die Vergabe der Program-Change-Nummern ist etwas unglücklich gelöst. Hier wäre eine ständige

Übersicht pro Entry, Seite etc. von Vorteil. Lobend muß ich erwähnen, daß zu jedem Song ein Notepad abgespeichert werden kann.

Der ein oder andere Musiker wird sicherlich auch in Erwägung ziehen, Freestyle auf der Bühne einzusetzen. Wer zu Hause eine entsprechende Musikanlage zur Verfügung hat und einen Begleitautomaten braucht, sollte sich überlegen, ob er nicht lieber ca. 300,-DM für Freestyle ausgibt, anstatt 3000,-DM für ein Rhythmus-Keyboard. Das Handbuch ist nicht nur in einem schönen Etui verpackt, sondern zudem auch übersichtlich aufgebaut und jeder Problemsituation gewachsen. Die 32 Styles auf der mitgelieferten Diskette sind von hervorragender Qualität. Ich hoffe, wir dürfen auf ein Update gespannt sein. Alles, was das Herz begehrt, ist vorhanden und steht zur Weiterverarbeitung bereit.

DVPI Session-Partner - Keine Ideen mehr? Überlassen wir es dem Zufall.

Welcher Computer-Musiker kennt es nicht - das Gefühl, alles richtig in den Sequenzer eingespielt zu haben, und doch klingt alles anders als es soll? Der Groove scheint zu stimmen, das Schlagzeug hämmert ganz toll, und dennoch klingt alles wie aus der Retorte. Hier und da ein Delay, ein paar Spuren kopiert und mit einer Transponierung versehen, die Sounds verändert, und alles klingt tierisch nach Kartoffelbrei. Hat man erst einmal ein paar Stunden vor seinem Computer verbracht, so läßt, wenn wir mal ehrlich sind, die Konzentration erheblich nach. Große Aktionen werden in der Regel dann nicht mehr von Erfolg gekrönt, und man beginnt am nächsten Tag von vorne. Es wird auch wohl nicht vorkommen, daß jeder gleich die richtigen Ideen hat und jeden Tag einen Top-40-Hit komponiert. Mir fällt jedenfalls nicht immer auf Anhieb das Richtige ein. Es wäre doch schön, wenn wir jemanden beauftragen könnten, für uns einen Song zu generieren. Dies soll „Session Partner" für uns erledigen. Ob das funktioniert, wird der nachfolgende Testbericht zeigen.

Das mir vorliegende Programm wird mit einer deutschen Bedienungsanleitung, 2 Disketten und einem Dongle geliefert. Auf den Disketten befinden sich das Programm und eine Demo. Das Dongle wird ausnahmsweise mal nicht in den ROM-Port des ATARI gesteckt, sondern in den noch freien Joystickport. Ich finde dies eine nette Geste, denn der ROM-Port ist ohnehin schon arg in Mitleidenschaft gezogen.

Vor dem Starten sollte sichergestellt sein, daß sich absolut keine Accessories im Speicher befinden und auch nicht automatisch hinzugeladen werden. Welche Möglichkeiten dieses Programm bietet, läßt schon der benötigte Arbeitsspeicher von mindestens 1 Megabyte erahnen. Session Partner nimmt davon satte 950 KB in Beschlag. Für 2 komplette Stücke bleibt also kein Platz. Wer einen Rechner mit mehr als 1 MB Arbeitsspeicher sein eigen nennen kann, ist hier auf jeden Fall im Vorteil.

Die Idee

Session Partner soll dem Benutzer die Möglichkeit geben, ohne große Vorkenntnisse und Programmiererfahrung eigene Musikstücke zu komponieren. Es werden 13 Musiker als eine komplette Band bereitgestellt, und jeder tut sein Bestes, um einen neuen Song zu erstellen. Dies geschieht entweder ganz automatisch oder nach veränderbaren Grundeinstellungen. Die so erzeugten Stücke können als MIDI-File abgespeichert und als Sequenzersong weiterverarbeitet werden. An Ideen kann es nun nicht mehr mangeln, da jederzeit ein neuer Groove oder Baß erstellt werden kann. Dies ist sehr einfach und wird keine Probleme bereiten, nur eine Übersicht, was denn nun eben gerade geschehen ist, gibt es nicht. Hier muß man sich ganz und gar auf das eigene Gehör verlassen. Session Partner generiert mit Einschränkung auch eigene Melodien, so daß eine MIDI-Thru-Funktion weggelassen worden ist. Dies ist sicherlich nicht die beste Lösung, erspart aber den Anschluß eines Masterkeyboards.

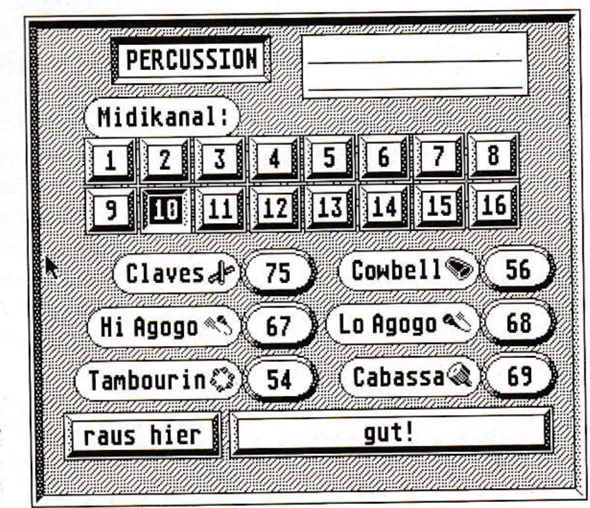
Equipment

Wer im Besitz mehrerer Expander ist, kann Session Partner so richtig aus der Reserve locken und in den Genuß aller implementierten Funktionen kommen. Es sollten 13 Klangmodule über verschiedene MIDI-Kanäle ansprechbar und auch separat in der Lautstärke regelbar sein. Den Rest besorgt Session Partner.

Voreinstellungen

Nach dem Programmstart begibt man sich als erstes daran, das MIDI-Setup an sein eigenes Equipment anzupassen. Hier ist die komplette Band vertreten. Niemand ist krank oder muß gerade mal schnell zur Toilette. Alle warten nur darauf, zum Einsatz zu kommen.

Jedem „Musiker" kann ein eigener MIDI-Kanal zugeordnet werden. Die Schlagzeuginstrumente, die normalerweise komplett über nur einen Kanal angesteuert werden können, sind hier noch einmal in Gruppen unterteilt. Somit ergibt sich die Möglichkeit, nicht nur die MIDI-Notennummern festzulegen, sondern auch verschiedene Percussion-MIDI-Kanäle auszuwählen. Es bleibt jedoch die Möglichkeit, durch Festlegen der Notennummern und Einstellen gleicher MIDI-Kanäle in gewohnter Manier zu arbeiten. Die reinen Melodieinstrumente wie zum Beispiel Baß, Orgel und Rhythmusgitarre



Das MIDI-SETUP-Menü bei Anwahl der Percussion-Instrumente

Geliftet



Fristeten frühere Versionen des REPRO STUDIO ST eher das Dasein eines Mauerblümchens, so bringt Trade It ihr Programm nun in einer 'Pro' (für Professional)-Version mit erheblich erweitertem Funktionsumfang auf den Markt.

REPRO STUDIOs neue Kleider

Im Verlauf des Tests zeigte sich, in wieweit es Trade It gelungen ist, mit ihrem Programm in die Spitze der Bildverarbeiter auf dem Atari ST/TT vorzustoßen und ob das Kürzel 'Pro' die Interpretation 'PROfessionelles PROgramm' oder eher 'PROblematisches PROdukt' verdient. Soviel vorweg: Diejenigen unter uns, die schon frühere Versionen des Repro Studios kennen, werden angenehm überrascht sein.

Neues Outfit

Das Programm wurde optisch und programmtechnisch komplett überarbeitet. Dieses war auch sehr nötig; fand man doch in älteren Programmversionen Funktionen wie z.B. 'Linie zeichnen' noch in den Pull-Down-Menüs(!), welche nun in der neuen Version (im folgenden RSTpro genannt) neben globalen Voreinstellungen sämtliche Funktionen zum Datei-Handling sowie zur Bilddatenein- und -ausgabe verwalten. Übersichtlich auf fünf Modulblöcke verteilt, befinden sich jetzt die Werkzeuge für folgende Funktionsbereiche: Zeichnen, Block, Retusche/Filter, Graustufenmanipulation und Vektorbearbeitung. Die Icons - in 'augenfreundlicher' Größe - sind zumeist recht aussagekräftig, so daß man sich im Programm recht schnell zu Hause fühlt. Im unteren Bereich eines jeden Moduls hat der Benutzer zu jeder Zeit Zugriff auf die Graustufenpalette, die Werkzeuggröße, die Regler für Helligkeit und Kontrast sowie auf einige weitere Funktionen und Schalter. Hier ist die Beschriftung jedoch etwas unglücklich geraten; so stehen z.B. die Kürzel 'VP' für 'Vektorobjekt als Zeichenpfad' und 'V' für 'Vektorobjekt in Bitmap'.

Die Dialogboxen wurden optisch auf den Stand der Zeit gebracht und lassen sich sehr angenehm bedienen; der Anwender nimmt fast sämtliche Einstellungen mittels Radio-Buttons, Check- und Pop-Up-Boxen vor.

```
Datei
 Neu ...              ^N
 Öffnen ...           ^O
 Schließen ...        ^U
 ---------------------
 Sichern              ^S
 Sichern als ...      ^M
 ---------------------
 Importieren ...      ^I
 Exportieren ...      ^E
 ---------------------
 Drucken ...          ^P
 Modul ...            ^T
 ---------------------
 Ende ...             ^Q

Optionen
 Fnkt. ausweiten      ^A
 Fnkt. gefüllt        ^G
 ---------------------
 Bild in Puffer ...
 Bild justieren ...   ^J
 Ausschnitt als Bild
 UNDO aktiviert       ^Z
 ---------------------
 Maske laden
 Maske speichern
 Block laden
 Block speichern
 ---------------------
 Voreinstellung ...   ^W
 Info ...
 Menü Hilfe ...

Scanner
 Prescan
 Prescan löschen
 ---------------------
 Seite scannen
 Ausschnitt scannen   ^R
 ---------------------
 Ausschnitt wählen ...
 Alle Ausschnitte
 Neue Attribute ...
 ---------------------
 Vollbild             ^V
```

Abb. 1: Die Pull-Down-Menüs von REPRO STUDIO ST pro

Der Datenaustausch mit anderen Programmen gestaltet sich problemlos: RSTpro im- und exportiert die wichtigsten auf dem ST gängigen Grafikformate, wie z.B. TIFF und Gem Image. Das RSTpro-eigene RPS-Format, welches neben dem Halbtonbild auch die verwendeten Masken, Vektorgrafiken und Paletten enthält, wird wahrscheinlich schon bald direkt in den Calamus SL einzulesen sein. Außerdem lassen sich Calamus-Vektorgrafiken in geöffnete Fenster importieren, weiterverarbeiten und auch wieder im CVG-Format exportieren. Als Eingabegeräte kommen in erster Linie Scanner zum Einsatz; angefangen von den leistungsstarken firmeneigenen Handy-Scannern bis hin zu hochwertigen Flachbett-Scannern diverser Anbieter (Epson, Logitech, Sharp...). Die Druckausgabe erfolgt wahlweise über Laser- oder Nadeldrucker (ATARI Laser, HP Laserjet, NEC...); auch für die Farbsublimationsdrucker von Mitsubishi sind Druckertreiber erhältlich. Vor dem Ausdruck läßt sich über 4 sog. LUT-Diagramme der Tonwertzuwachs frei definieren. Diese Option wirkt sich nicht auf den Bildschirm aus, sondern wird erst beim Ausdrucken bzw. beim Rastern und Speichern des Bildes aktiv und dient somit der Druckoptimierung.

Das Handbuch ist vom Verhältnis Inhalt/Umfang als ausgewogen zu bezeichnen. Die diversen Funktionen werden ausreichend detailliert erläutert und durch nützliche Hinweise und Tips aufgelokkert; allenfalls die Abbildungen sind manchmal etwas zu klein geraten. Angefangen bei der Installation und einer Quick-Referenz, gefolgt von separaten Erläuterungen zum Monochrom- bzw. Halb-

tonmodus bis hin zum Glossar und einem ausführlichen Index erfährt der Benutzer alles Wissenswerte über Programm und Peripherie.

Werkzeugkasten

RSTpro ist - für ein EBV-Programm - sehr reichhaltig mit Werkzeugen ausgestattet, die eher typisch für ein monochrom arbeitendes Grafikprogramm sind: so findet man neben Freihandlinien und Polygonzügen auch Ovale, Rauten, Parallelogramme und sogar Bézier-Kurven. Diese Zeichenfunktionen haben jedoch zwei zusätzliche Aufgaben: einerseits lassen sie sich in Graustufenbildern als normale Zeichenwerkzeuge benutzen, andererseits kann man sie aber auch zum Erzeugen von Masken einsetzen. Sogar die Sprühdose läßt sich zur Maskengenerierung heranziehen, wodurch z.B. interessante Überblendeffekte beim Kombinieren von Scans möglich sind. Ist der Schalter 'Funktionen ausweiten' aktiv, scrollt der Bildschirm bei Erreichen des Fensterrandes automatisch weiter, wodurch der Aktionsradius der Zeichenwerkzeuge nicht nur auf den sichtbaren Bildschirmausschnitt begrenzt ist. Um auch Bilddateien bearbeiten zu können, die nicht mehr in den Arbeitsspeicher des Rechners passen, lagert RSTpro Bildteile, die gerade nicht benötigt werden, virtuell auf Festplatte aus. Im Extremfall könnten so auch Grafiken bearbeitet werden, die 100 MB und größer sind, natürlich genügend Plattenkapazität vorausgesetzt.

RSTpro stellt einen UNDO-Puffer bereit, in den jederzeit durch Druck auf die Space-Taste der aktuelle Bildschirminhalt 'hineingesichert' werden kann, um bei Bedarf das ganze Bild oder mittels des 'Teil-Undo'-Werkzeuges auch beliebige Teile des Bildes zu restaurieren. Durch wiederholtes Betätigen der Undo-Taste läßt sich zwischen dem aktuellen Bild und dem Undo-Puffer hin und her schalten. So lassen sich das 'Original' und die manipulierte Version bequem miteinander vergleichen. Übrigens existiert auch für die Maske ein Undo-Puffer, welcher mittels Shift-Space aufgefrischt wird. So können auch Fehler bei der Maskenerstellung schnell wieder behoben werden.

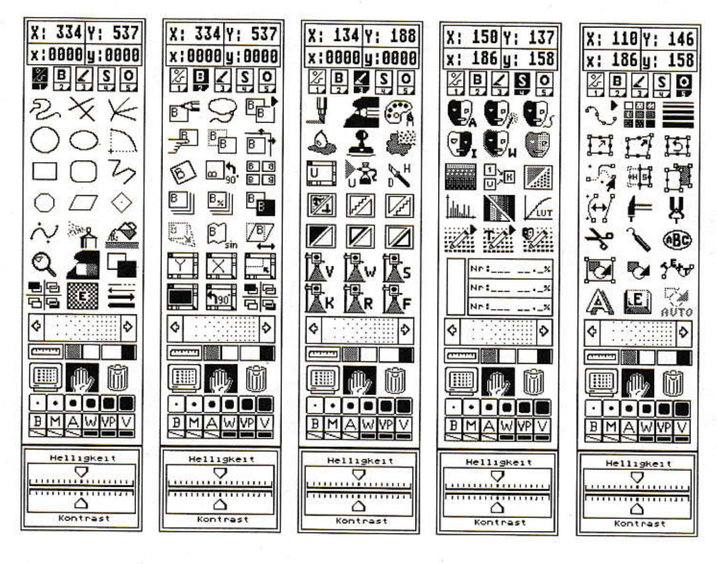


Abb. 2: Die diversen Funktionen und Werkzeuge wurden übersichtlich auf fünf Modulblöcke verteilt.

Blockiert...

RSTpro unterscheidet rechteckige Blöcke und sog. Lassos. Während die normale Blockfunktion ausschließlich rechteckige Bildbereiche ausschneidet, kommt das Lasso in Verbindung mit der Maske zum Einsatz und ermöglicht das Ausschneiden unregelmäßig geformter Blöcke. Einmal definiert, lassen sich die Blöcke auf vielfältige Art und Weise manipulieren; z.B. drehen um beliebige Winkel, spiegeln, skalieren, invertieren und verzerren. Sogar die Projektion auf eine frei editierbare Sinuskurve ist möglich. In Kombination mit der Verzerrfunktion lassen sich hiermit z.B. im Wind flatternde Fahnen oder - wie in der Beispielmontage - gewelltes Papier spielend realisieren. Bildteile, die sich im Blockpuffer befinden, lassen sich natürlich auch in andere Fenster einmontieren; die Transparenz läßt sich vorher stufenlos einstellen. War der Block als Lasso definiert, werden seine Umrisse - bei aktiviertem Masken-Weichzeichner - beim Einsetzen ins Zielbild automatisch geglättet und dem Zielbild angepaßt.

Sämtliche zur Bildbearbeitung notwendigen Retuschewerkzeuge und Filter sind in einem gemeinsamen Modul untergebracht. Neben dem Schwamm, der dazu dient, Graustufenübergänge zu verwischen und harte Konturen und Kontraste aufzuweichen, sind hier ein Spachtel zum Aufhellen bzw. Verdunkeln von Bildteilen, ein Marker und diverse andere Malwerkzeuge vorhanden. Mit dem Finger-Werkzeug lassen sich Grauwerte in andere Bildbereiche 'hineinschmieren' und so gezielt Feinheiten herausarbeiten. Für ziemlich überflüssig halte ich den Wasserpinsel. Dessen Verhalten erinnert zwar durchaus an das reale Vorbild, ein sinnvolles Einsatzgebiet für dieses Werkzeug kann ich mir jedoch beim besten Willen nicht vorstellen. Als sehr hilfreich dagegen erweist sich das sog. 'partielle Undo': mit diesem Werkzeug lassen sich Manipulationen an der Grafik in beliebigem Umfang wieder rückgängig machen; vorausgesetzt, man hat den Undo-Puffer, auf dessen Bildinformationen hierbei zurückgegriffen wird, rechtzeitig aktualisiert. Demselben Zweck dient auch die Undo-Sprühdose, hier jedoch mit einem Airbrush-Effekt.

Diverse Filterfunktionen ermöglichen globale bzw. durch Block und/oder Maske lokal begrenzte Bildmanipulationen wie z.B. Weichzeichnen, Verwaschen, Schärfen, Konturieren oder Aufrauhen. Besondere Erwähnung verdient hier der sog. 'freie Filter'. Mit seiner Hilfe ist es z.B. möglich, Reliefeffekte zu erzeugen. Hierbei scheinen dunkle Flächen 'erhaben' aus dem Bild herauszutreten, wie man in dem

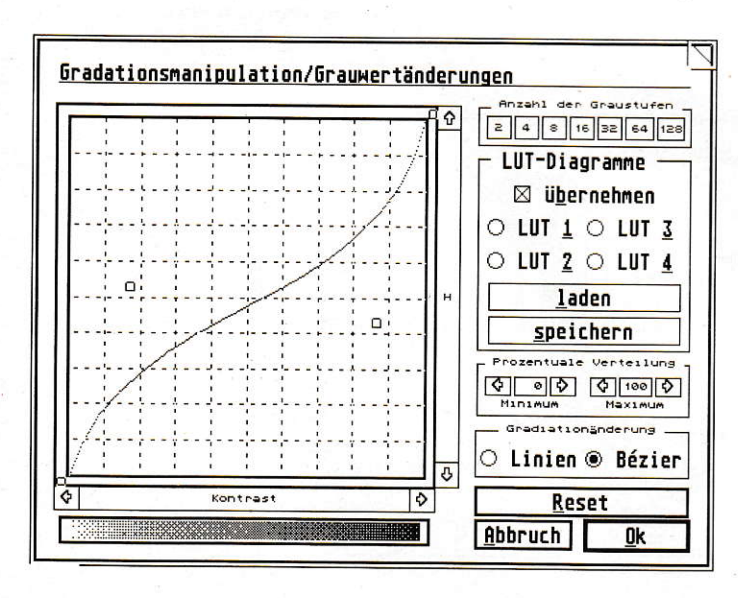


Abb. 3: Gradationsveränderungen lassen sich entweder freihändig oder mittels Bézier-Kurven bestimmen.

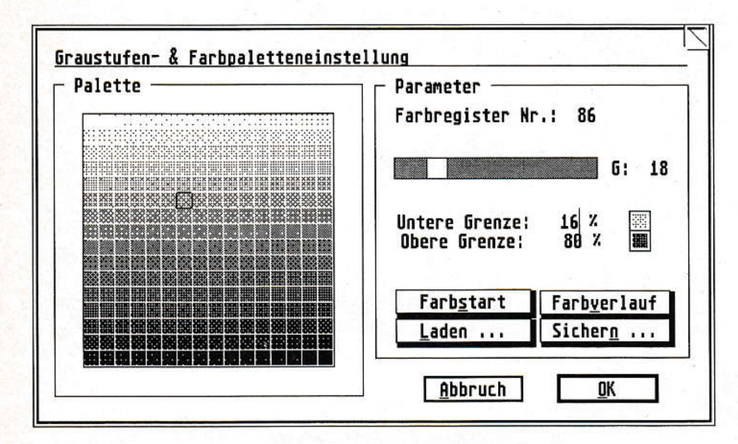


Abb. 4: Auswahl aus 256 Graustufen bzw. Farben

zweiten Beispielbild (RSTpro-Schriftzug) gut erkennt. Auch die Monochromfilter sollten nicht unerwähnt bleiben: Mit ihrer Hilfe lassen sich Bitmap-Grafiken z.B. glätten, konturieren und die Linienstärken ausdünnen bzw. verstärken. Sämtliche Filterfunktionen arbeiten erfreulich schnell. Dasselbe gilt übrigens auch für den Bildschirmaufbau, der ebenfalls sehr flott vonstatten geht.

Maskiert...

Um Bildmanipulationen auf bestimmte Graustufen bzw. Bildbereiche zu begrenzen, bietet RSTpro diverse Möglichkeiten, Masken zu erzeugen. Beim Einsatz einer Grafikkarte erscheint die Maske übrigens transparent rot, wodurch auch die Bildteile noch gut erkennbar sind, die von der Maske bedeckt werden. Neben der Möglichkeit, die Maske manuell bzw. unter Zuhilfenahme der Monochromwerkzeuge zu zeichnen, bietet RSTpro auch 'automatische Masken' an. So z.B. die 'Maske durch Zauberstab', die benachbarte Graustufen maskieren soll. Diese Funktion sollte man jedoch schnell wieder vergessen, da sie extrem langsam und uneffektiv arbeitet. Es geht jedoch auch anders: so z.B. mit der Funktion 'Maske zwischen Graustufen'. Nach Festlegen eines Graustufenbereiches - dies geschieht durch Überstreichen mit der Maus - werden die in diesem Bereich liegenden Graustufen automatisch maskiert. Sollten noch Grauwerte fehlen, wiederholt man ganz einfach diese Prozedur; die neugefundenen Maskenteile werden der 'alten' Maske dann hinzugefügt. Um im Beispielbild den Oberkörper vor dem Hintergrundverlauf zu schützen, wählte ich eine Kombination aus manuellem und automatischem Maskieren, was insgesamt sehr zügig vonstatten ging.

Im gleichen Modul befinden sich auch die sog. LUT-Funktionen zur Manipulation der Gradationskurve. Bei Einsatz einer Grafikkarte sieht man deren Änderungen in Echtzeit auf dem Bildschirm; das gleiche gilt für Änderungen von Helligkeit und Kontrast! Dieses Feature, welches man schon bald nicht mehr missen möchte, ermöglicht eine unmittelbare Ergebniskontrolle. REPRO STUDIO ST pro arbeitet übrigens auch in der 'Graustufen-Version' mit Farbe, allerdings nur begrenzt. In der Graustufen- bzw. Farbpalette lassen sich über drei RGB-Regler beliebige der 255 Grauwerte mit Farben belegen. Wenn auch diese Option aus RSTpro noch lange kein 'Farbretusche-Programm' macht, so ist sie zumindest für Schmuckfarben und farbige Beschriftungen sehr gut zu gebrauchen.

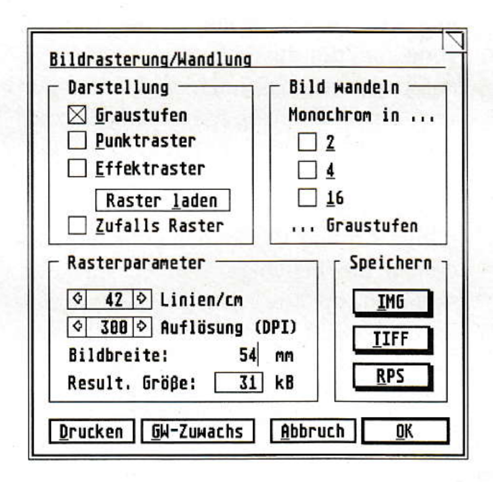


Abb. 5: Die Wandlungs- und Rastermöglichkeiten für die Bilddatenausgabe

Auch Verläufe sind für RSTpro kein Problem. Neben vertikalen und horizontalen Verläufen beherrscht RSTpro auch Kreisverläufe. Da die Graustufen der Verläufe nicht automatisch verrauschen, können unter ungünstigen Umständen Stufen sichtbar werden. Daher sollte man die Verläufe im Anschluß mit einem Unschärfe-Filter behandeln. Als sehr effektvoll erweist sich die Wiederholfunktion der linearen Verläufe; hierbei legt man nach dem Umriß des Verlaufs den Abstand fest, mit dem sich der Verlauf bis zum Ende wiederholt. So auch geschehen in meiner Beispiel-Grafik, wo der Effekt sehr an ein Fensterrollo erinnert. Leider lassen sich die Verläufe nicht in einem freien Winkel erzeugen; diese Funktion wird einem späteren Update vorbehalten sein (ebenso wie die freie Definition des Verlaufsursprungs bei Radialverläufen).

Auf Vektor-Pfaden...

Importierte Vektorgrafiken können manipuliert bzw. neue Grafiken erstellt werden. Hierzu lassen sich über ein kleines Pop-Up-Menü geometrische Grundfiguren erzeugen oder freie Linien und Bézier-Kurven zeichnen. Neben den Standard-Vektorfunktionen wie z.B. Punkte setzen/löschen und Pfade auftrennen/verbinden lassen sich Linien und Bézier-Kurven tauschen sowie Gruppen bilden und auflösen. Jedem Vektorobjekt kann eine beliebige Linienstärke sowie eine der 256 Graustufen in Verbindung mit einem Füllmuster zugeordnet werden. Leider werden die Objekte nach jeder Manipulation komplett neu gezeichnet. Da das Vektormodul ohnehin etwas 'gemächlich' zur Sache geht, sind Wartezeiten leider die Regel. Das Vektormodul bietet weiterhin die meines Wissens einzigartige Funktion, bis zu vier sog. 'freie' (also nicht lizensierte) Calamus-Fonts zu laden und (nicht nur) zur Titelgenerierung zu verwenden. In einem separaten Fenster verfolgt man in

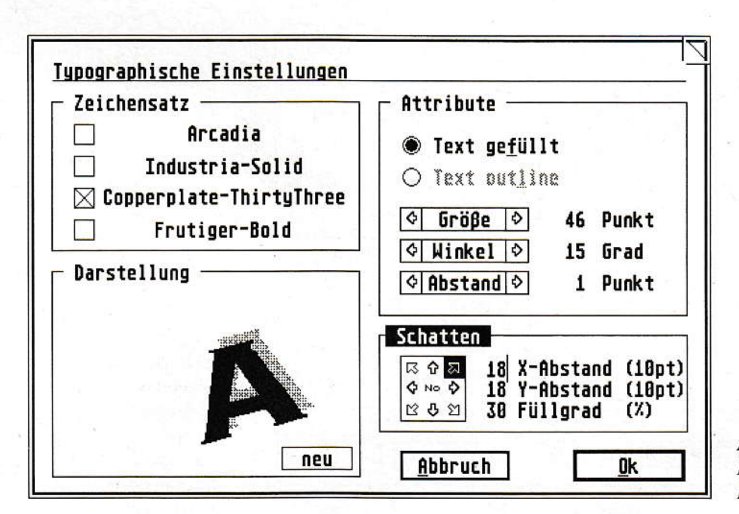


Abb. 6: Auch Beschriftungen sind in RSTpro kein Problem

Echtzeit das Vergrößern und Drehen des jeweiligen Fonts; selbst ein Schatten läßt sich in beliebiger Richtung setzen. Nach dem Festlegen der Textattribute wird dann der eigentliche Text in einem leider sehr spartanisch ausgefallenen Zeileneditor eigegeben, der jedoch für Headlines und kleinere Texteinblendungen ausreicht. Leider ist es noch nicht möglich, Sonderzeichen einzugeben, die nicht auf der Tastatur liegen. Nach Betätigen der Escape-Taste erscheint der Text dann im Grafikfenster, wo er weiter 'behandelt' werden kann. Trommel- und Kugelprojektionen sowie freies Verzerren seien hier nur zwei Stichworte. Auf diese Weise entstand auch der 'Graffiti'-Schriftzug in dem Beispielbild, den ich halbtransparent auf die Mauerstruktur gelegt habe. Es ist ebenfalls möglich, Zeichenwerkzeuge und Maskierstifte entlang von Vektor(text)pfaden laufen zu lassen, was weitere kreative Möglichkeiten eröffnet. Dank der (begrenzten) Farbmöglichkeiten von RSTpro lassen sich übrigens auch farbige Vektorgrafiken leicht realisieren und effektvoll mit Graustufenbildern kombinieren.

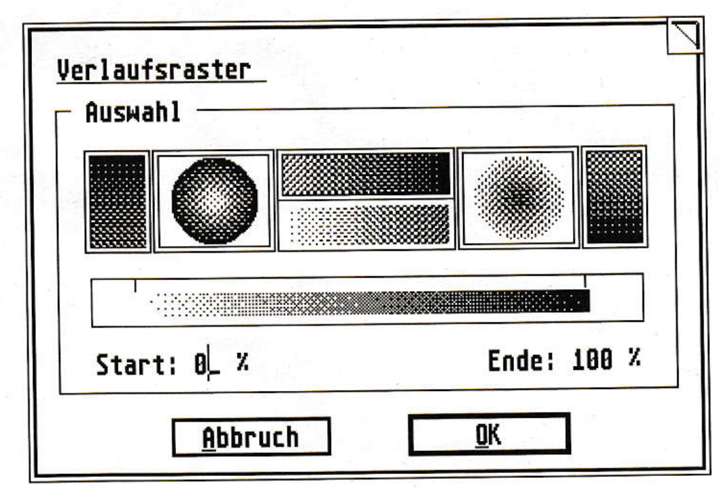


Abb. 7: Die Verlaufsrasterdialogbox vonRSTpro

Ausblick und Fazit

Trade It plant zur CeBIT '92 die Fertigstellung einer 24-Bit-Farbversion des REPRO STUDIOs, die um spezielle Funktionen zur Farbbildverarbeitung erweitert sein wird. Neben Geschwindigkeitsoptimierungen wird auch an einem Vektorisierungsmodul sowie an diversen externen Modulen getüftelt; denkbar wäre hier z.B. ein Vermessungsmodul für Luftaufnahmen. Realität ist bereits ein separates 'PostScript-Modul'; zum Preis von 498,- DM ermöglicht dieses die direkte Bildformat-Wandlung in PostScript, GEM3 und viele andere Formate.

Wer momentan sowohl gescannte Graustufen- als auch Monochrombilder weiterverarbeiten und für die Übernahme in einen Publisher bzw. die Direktausgabe auf einen Drucker aufbereiten möchte, dem kann 'REPRO STUDIO ST pro' in der aktuellen Version guten Gewissens empfohlen werden. Kleinere Schwächen (z.B. Geschwindigkeit des Vektormoduls) werden durch andere Features (z.B. die flexible Blockbearbeitung) ausgeglichen. Der Preis scheint mir mit 998,- DM nicht zu hoch angesetzt. Will man am Bildschirm mit 'echten' Graustufen arbeiten, muß man jedoch noch weitere 98,- DM für einen GEM-Grafikkartentreiber bzw. 198,- DM für einen speziellen TT-Bildschirmtreiber hinzurechnen. Letztgenannter hat es aber in sich: im 16-Farben-Modus des TT (Auflösung 640 x 480) stellt RSTpro dank trickreicher Programmierung 256 Graustufen dar; selbst die Maske erscheint noch transparent rot! Als Ausgleich für diese zusätzlichen Kosten liefert Trade It auf den Programmdisketten bereits diverse Scanner- und Druckertreiber mit, so daß einem sofortigen Start in die Welt der elektronischen Bildverarbeitung nichts im Wege steht.

Matthias Ficht

Bezugsquelle:
Trade It
Arheilger Weg 6
W-6101 Roßdorf
Tel. 06154/9037

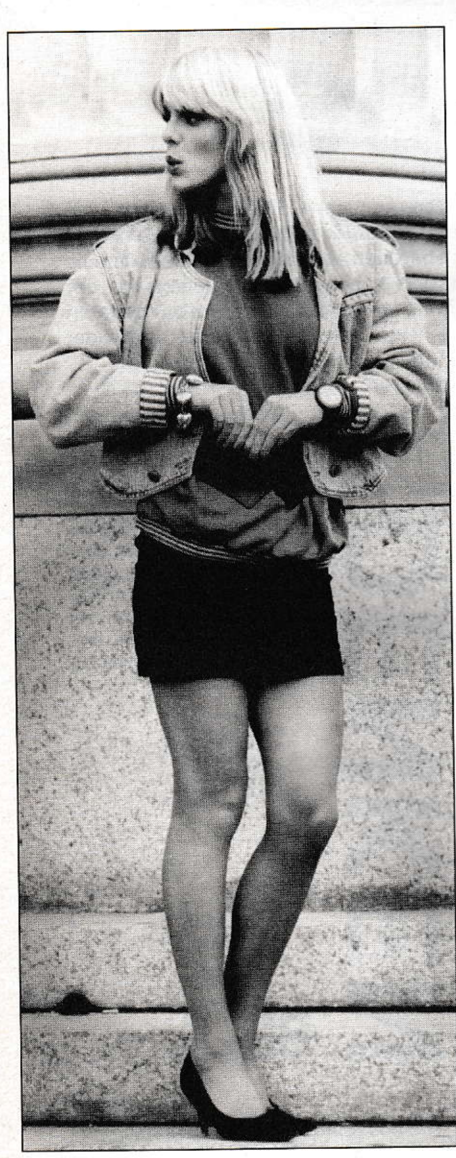
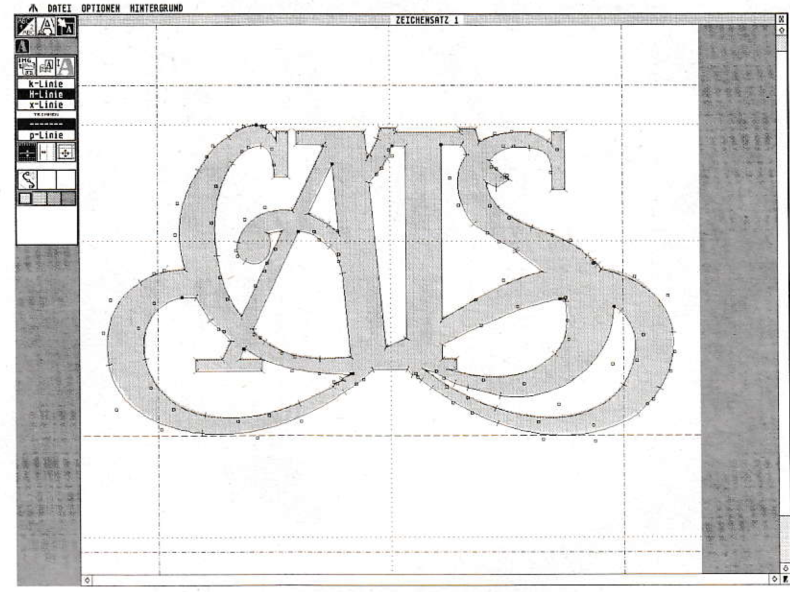


Abb. 8A: Vorher...



Abb. 8B: ...nachher!

Typo-Magie



Type Art - Ein Präzisionswerkzeug für alle Vektor-Fälle

Vielleicht war am Anfang ja das Wort, und vielleicht entwickelte sich unsere Sprache ja, wie Linguisten heute vermuten, auch durch „der Hände Arbeit". Daß dieses zumindest für die geschriebene Sprache, die Schrift, zutreffen mag, wird für jeden sofort nachvollziehbar, der sich schon einmal gestalterisch um eine solche bemüht hat. In der gesamten Geschichte der Schriftkunst gab es noch nie gleichzeitig so viele verschiedene und auch eigenständige Stile wie in unserem Jahrhundert.

Wie die gesprochene Sprache, befindet sich die Schrift jedoch in einer ständigen Entwicklung - neue Ideen schaffen neue Zusammenhänge, schaffen und bedingen auch neue Schriften. Eine „endgültige" Schrift wird es darum ebenso wenig geben wie eine vollständige Schriftensammlung in Ihrem Font-Ordner. Diese „typografische Inflation" spiegelt auch den Zeitgeist wider, wobei vor allem durch und in der Werbung die „Schrift an sich" zu einem Idiom, zu einer auch grafisch erfaßbaren Gleichung für ihren eigentlichen Inhalt wurde. Gerade im DTP-Bereich, dessen gestalterische Basis ja in erster Linie „Schrift" ist, erfordert diese Entwicklung ein exzellentes Werkzeug, um gerade benötigte Zeichen korrekt erstellen, oder zumindest die schon vorhandenen dem aktuellen Bedarf entsprechend modifizieren zu können. Ein faszinierendes Werkzeug zur vektoriellen Schriftgestaltung und digitalen Reinzeichnung liegt den Desktop Publishern nun mit dem Font-Editor TYPE ART aus dem Calamus-Haus DMC vor.

Wozu aber einen Font-Editor, wo doch inzwischen jede mögliche Schrift von den verschiedensten Anbietern erhältlich ist? Auch wer so denkt und keine weiteren Ambitionen bzgl. einer eigenen Schriftgestaltung hat, wird mit Type Art zaubern können: Vorhandene Fonts lassen sich en bloc vielfältig modifizieren, mit anderen mischen und in manchen Fällen auch erheblich verbessern, wie wir später noch sehen werden. Neue Schriften (z.B. „Kapitälchen") lassen sich aus beliebigen Fonts schnell erstellen und als neuer Font abspeichern. Fehlende Sonderzeichen oder Umlaute (ä, ü, ö) werden für den gesamten Font automatisch generiert, und sogar „zusammengestückelte" Fonts oder Grafiken können auf Knopfdruck in saubere Outlines verwandelt werden! Und das ist längst noch nicht alles.

Fliegende Boxen und freie Formen

Type Art ist ein eigenständiges Programm und nicht als Calamus SL-Modul zu laden, wie man vielleicht vermuten könnte. Es läuft ab 1 MB-Speicher auf allen ST- und TT-Rechnern. Nach dem Programmstart präsentiert es sich denn auch in einem Calamus-ähnlichen Layout: über die Menüleiste sind alle Zugriffe auf Speichermedien, Im- und Exporttreiber, globale Rechenoperationen usw. untergebracht. In der Kopfzeile lassen sich Arbeitsbereiche anwählen, über die sich die gewünschten Werkzeuge aus umfangreichen Bearbeitungsgruppen heraussuchen lassen. Insgesamt teilt sich Type Art in drei Arbeitsbereiche auf: die Zeichensatzbearbeitung, die Erstellung von Vektorzeichen und Grafiken sowie die Rasterbildverarbeitung nebst automatischer Vektorisierung. Hinter jeder dieser Gruppen liegen weitere Befehlsfelder mit zusätzlichen, teilweise ganz schön umfangreichen Submenüs. Den klug konzipierten, hierarchisch angelegten Befehlseingaben ist es hier zu verdanken, daß bei aller Vielfalt der Werkzeuge das gesamte Programm einen außerordentlich bedienerfreundlichen Eindruck macht. Ein Eindruck, der sich auch bei der Arbeit mit dem beiliegenden Handbuch bestätigt: Gut die Hälfte wird von einem Tutorial bestimmt, in dem Schritt für Schritt und anhand vieler Beispiele mit der Arbeitsweise in Type Art vertraut gemacht wird. Auch auf kleine typografische Besonderheiten beim Erstellen von Vektor-Fonts wird eingangen, was für manchen den Einstieg in diesen Bereich etwas leichter machen wird.

Die Type-Art-Arbeitsfläche paßt sich nach dem Start der Größe des benutzten Monitors an, wobei auch in höheren Zoom-Stufen die GEM-üblichen Scroll-Balken ein von anderen Programmen gewohntes Arbeiten ermöglichen. Schon das Anwäh-

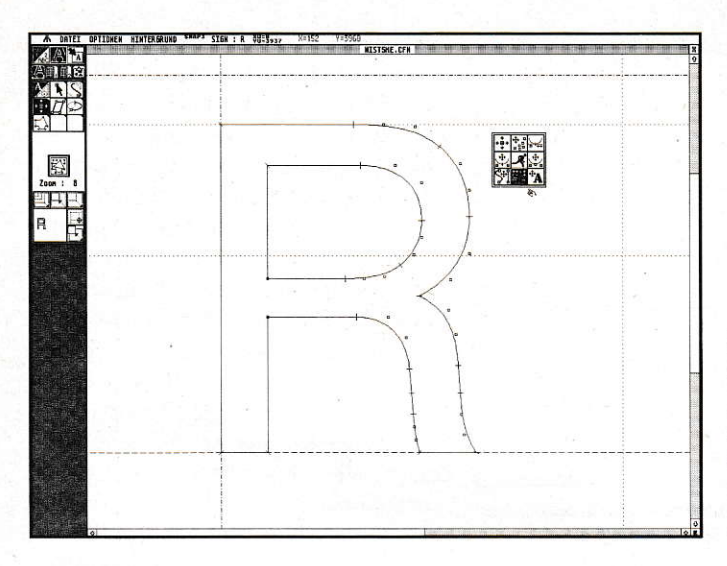


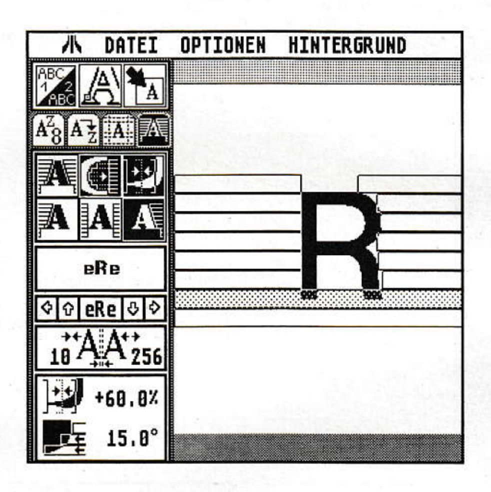
Bild 1: Das Type Art-Desktop. Die Werkzeuge der einzelnen Arbeitsmodi sind in Pop-Ups zusammengefaßt, die auf Klick an der Mausposition erscheinen.

Tatsächlich entwickelte sich unsere Sprache
Tatsächlich entwickelte sich unsere Sprache

Bild 2: Durch Feineinstellungen zur Zeichenerkennung liefert Type Art eine bessere Ausgangsbasis für die dann folgende Handarbeit (Zeile 1), als dieses bisher im reinen Stufen-Kerning möglich war (Zeile 2).

len eines der Icons im Bearbeitungsfenster zeigt jedoch eine der Besonderheiten in der Bedienung dieses Editors. Hinter jedem Icon verbirgt sich ein Pop-Up-Menü, das die zum jeweiligen Bearbeitungsmodus gehörenden Werkzeuge beinhaltet. Dieses Menü kann auf Mausklick überall im Fenster aufgerufen werden und erscheint umgehend an der aktuellen Mausposition. Extrem gewöhnungsbedürftig, aber auch genauso nützlich für die Arbeit am Großbildschirm. Da der reibungslose Umgang mit den Icons aber voraussetzt, daß deren Bedeutung auch bekannt ist, können für den ersten Einstieg Online-Hilfen gewählt werden, die zu jeder angewählten Funktion noch vor der Ausführung eine mehrzeilige Erklärung einblenden. Neben allen für Vektorzeichenarbeiten relevanten Werkzeugen können in den Pop-Ups auch einige geometrische Objekte (Dreieck, Ellipse usw.) direkt angewählt werden, die sich teilweise auch gleich mit einer zusätzlichen Outline konstruieren lassen.

In der Darstellung und Bearbeitung der Vektorobjekte geht Type Art anwenderfreundliche und zum Teil auch völlig neue Wege. Wird ein Stützpunkt oder eine Tangente auch nur „berührt", ertönt ein akustisches Signal(!), und die entsprechende Linie wird mit allen dazugehörigen Punkten hervorgehoben dargestellt. Liegen mehrere Punkte über- oder dicht beieinander, werden die einzelnen Punkte automatisch und nach einer einstellbaren Zeitvorgabe nacheinander selektiert. Ein Mausklick fixiert dann das gewünschte Segment für die Bearbeitung (Einfügen, Kopieren, Projektionen usw.). Ein absolut exaktes Arbeiten ist durch dieses Verfahren gewährleistet, ohne daß „versuchsweise" Stützpunkte verschoben werden müssen, um z.B. Zugehörigkeiten zu erkennen. Auch in der Bearbeitung von Hilfslinien hat Type Art etwas



Besonderes zu bieten: Hilfslinien werden nach Anwählen des entsprechenden Icons wie Vektorobjekte erstellt, mit allen hier zur Verfügung stehenden Werkzeugen. Es ist somit auch möglich, die konstruierten Hilfslinien als Vektorgrafik (was sie ja sind) abzuspeichern oder die Zeichen eines geladenen Fonts und auch importierte CVGs als Hilfslinien zu definieren. Diese werden beim Speichern eines Fonts im CFN-Format automatisch gesichert und bleiben so fürdie nächsten Sitzungen erhalten. Zur besseren Unterscheidung kann Punkten, Beziers, Linien und Tangenten auch eine Farbe zugeordnet werden, was bedeutet, daß Type Art auch in diesem Modus seine Dienste tut. In der aktuellen Version werden jedoch einige Grafikkarten noch nicht unterstützt, so z.B. die Coco/C 32 und C110 von Matrix; in den verschiedenen TT-Modi läuft Type Art jedoch einwandfrei.

Diese differenzierte Darstellung der Oberfläche und der zu bearbeitenden Objekte ist ohne Zweifel vorteilhaft, reicht aber für eine angemessene optische Kontrolle der erstellten Zeichen noch nicht aus. Type Art stellt Vektorobjekte, und in einem Font-Editor sind das nun einmal zu allererst Schriftobjekte, in der aktuellen Version nur als Outline-Pfad dar; eine Darstellung in gefüllter Form ist noch nicht möglich. Die Gesamterscheinung eines Zeichens läßt sich jedoch erst im Schwarzweißkontrast, sozusagen zwischen „innen und außen", korrekt beurteilen (ich denke hier beispielsweise an die Punzen beim „e" oder „o"). Wünschenswert wäre hier also eine auch gefüllte Darstellung von Vektorobjekten, die sich dann vielleicht sogar - wie in „Ikarus" oder auch im Calamus SL-Vektormodul - editieren ließe. Ein weiteres Problem tritt in diesem Zusammenhang fast zwangsläufig auf. Wird eine Vektorgrafik importiert, die in einer externen Software (Outline Art, Didot) schon mit Rasterungen versehen wurde, sind diese nach einer Bearbeitung in Type Art verschwunden bzw. werden durchgängig „schwarz" abgespeichert.

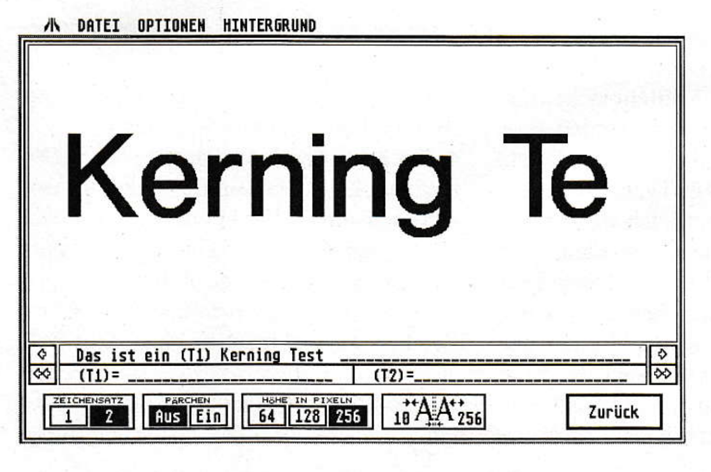


Bild 3a und b: Zur Kerning-Arbeit steht neben dem gut gefüllten Bearbeitungsfeld auch eine kleine Satzfunktion zur Verfügung, mit der ein Test-Text des gerade bearbeiteten Fonts in unterschiedlichen Größen dargestellt wird. Kerning Änderungen werden dabei sofort übernommen.

Font-Arbeit

Wie überhaupt im gesamten Programm, sind auch die Funktionen zur Font-Bearbeitung alternativ per Tastatur aufrufbar. Es genügt ein Klick auf die Tastatur, und das entsprechende Vektorzeichen des Fonts erscheint umgehend im Fenster. Gleich 2 Fonts können sich gleichzeitig im Speicher befinden, zwischen denen dann auch bei der Kerning-Bearbeitung hin- und hergeschaltet und beispielsweise verglichen werden kann. Geladen, bearbeitet und gespeichert werden können Vektorgrafiken (CVG) sowie alle Fonts im CFN-Format, auch serialisierte. CVGs können zudem wahleise ins Arbeitsfenster oder direkt ins Clipboard geladen werden. Im 2.0-Update wird es möglich sein, zusätzlich alle Type-1-Schriften der PC- und MAC-Welt zu laden und ins CFN-Format zu konvertieren. Auch die in schier unendlicher Zahl erhältlichen Signum-Zeichensätze können dann via Type Art als Vektor-Fonts des Calamus Licht und Schatten erblicken! Für die reine Font-Arbeit sind in Type Art vier Bearbeitungsfelder vorhanden:

1. Die Zeichensatzauswahl, eine Übersicht aller Zeichen des (oder der beiden) geladenen Fonts
2. Die Fontbearbeitung. In diesem Bearbeitungsfeld können ausgewählte Bereiche eines Fonts in einen zweiten kopiert und auch eventuell fehlende Sonderzeichen („Ö", „=" usw.) und Umlaute (ä, ü, ö) automatisch (!) erzeugt werden. Type Art benutzt zu diesem Zweck die schon vorhandenen Zeichen und generiert dann z.B. aus „.." und „A" ein „Ä". Genauso, wie man es auch in Handarbeit machen würde, geschieht es nun auf Knopfdruck. Mit den hier vorhandenen Funktionen können zudem, ohne daß man selbst zum Font-Schneider werden muß, aus vorhandenen Fonts gebrauchsfähige neue Fonts erstellt werden. Beispielsweise kann eine Kapitälchen-Schrift (nur Versalien, mit etwas größeren „Großbuchstaben") auch für nicht Schriftkundige schnell entwickelt und als neue Schrift gespeichert werden.
3. Die Einstellung der Ausrichtungslinien. Diese Linien können in Type Art für ein einzelnes Zeichen oder auch für den gesamten Font exakt gesetzt werden. Hier kann z.B. auch durch eine korrekte Einstellung des M-Squares - das sind die Begrenzungslinien, von denen alle Zeichen komplett umschlossen und nach denen im Calamus Zeichenhöhe und Zeilenabstand eines Textes errechnet werden - ein schon vorhandener, falsch eingestellter Font nachträglich neu berechnet werden.
4. Das Kerning-Menü. Par excellence. Die hier mögliche Präzision ist selbst in den automatisierten Funktionen erstaunlich. Für das saubere Kerning eines rohen Fonts müssen generell einige Arbeitstage eingeplant werden: die Hohlräume einzelner Zeichen werden manuell eingestellt (z.B „C", „K"), eventuell vorhandene Serifen sowieso, Rundungen eingezogen und die Unterschneidung problematischer Buchstabenpaare (z.B. „Ta") gesondert behandelt. Aufgelockert wird das Ganze nur durch viele Probeausdrucke in den verschiedensten Größen und Test-Texten. Das hört sich schwierig und zeitintensiv an, es ist auch so. Nicht, daß Type Art all diese Arbeiten auf Knopfdruck erledigt - oder doch? Nach Aussage der Handbuchautoren wurden für Type Art Programmroutinen entwikkelt, die das automatische Unterschneiden aller Zeichenkombinationen eines Zeichensatzes mit einer Zuverlässigkeitsquote von bis zu 98% ermöglichen. Wie kann so etwas bei einem mathematisch gar nicht richtig faßbaren Objekt wie der menschlichen Schrift möglich sein? Da in Handbüchern jedoch oft allerhand stehen kann, was auf dem Bildschirm dann partout nicht mehr nachvollziehbar ist, reizt so eine Aussage natürlich zu einer praktischen Überprüfung.

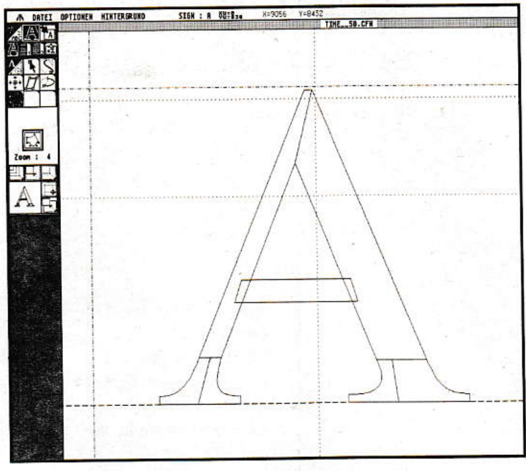
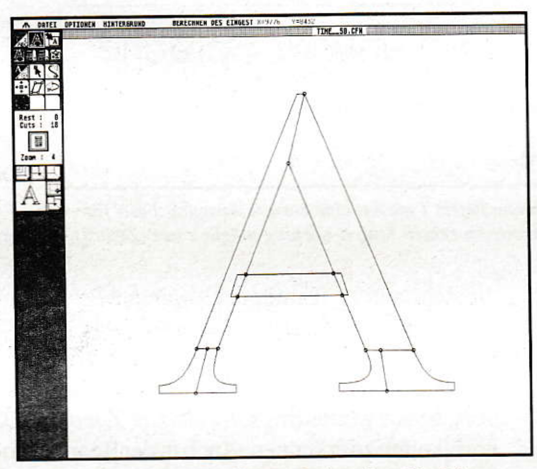
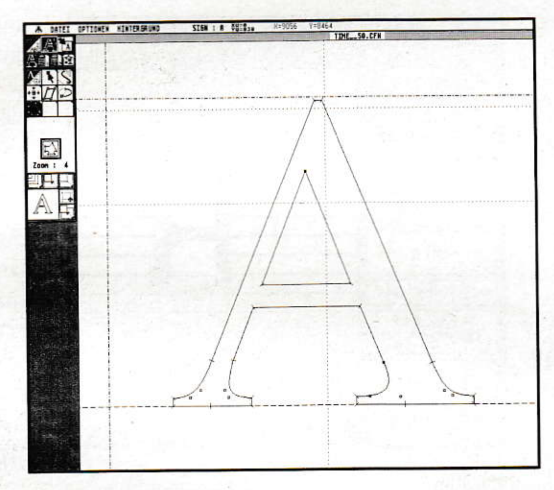


Bild 4a: Der kurze Weg zur sauberen Vektor-Outline: Das Zeichen eines Fonts, aus einzelnen Segmenten zusammengesetzt.



4b: Vom Programm werden nacheinander alle sich überschneidenden Linien markiert ...



4c: ... und gelöscht. Übrig bleibt ein sauberes Outline-Zeichen.

Zusätzlich zum in Font-Editoren üblichen Dickten-, Block- und Stufen-Kerning („Dickte" bezeichnet die absolute Breite, „Stufen-Kerning" beschreibt die konkreten Außenlinien eines Zeichens) bietet Type Art eine Feineinstellung zur Hohlraum-, Serifen- und Steigungserkennung der im Font verwendeten Zeichen an. Zu dieser können Prozentwerte eingegeben werden, nach denen beispielsweise die Hohlräume im „C" oder „K" oder auch Verjüngungen wie im „T" erkannt und berechnet werden. Auch Serifen können um einen frei einstellbaren Wert beschnitten und bei runden Zeichen die Kerning-Treppe etwas eingerückt werden. Die Berechnung des gesamten Zeichensatzes geschieht dann nach den einmal eingestellten Werten vollautomatisch und dauert auf dem TT im Schnitt weniger als 30 Sekunden (Test mit SWISS 50, 160 Zeichen belegt, 20 Sekunden)! Wie die Abbildung zeigt, liefert selbst die werkseitige Voreinstellung gute Ergebnisse. Man braucht nur einmal einen beliebigen Font in die beiden Font-Speicherplätze zu laden, und dann einen davon von Type Art nachkernen zu lassen. Der Vergleich zwischen beiden kann direkt in Type Art vorgenommen werden, wozu sich ein weiteres Fenster aus dem Kerning-Menü heraus öffnen läßt. In verschiedenen Darstellungsgrößen kann hier ein beliebiger Text eingegeben und auf Knopfdruck in den beiden vorhandenen Schriften angezeigt werden. Jede nachträgliche Kerning-Änderung einzelner Zeichen oder im Pärchen-Kerning-Modus wird sofort im Text realisiert.

Abrakadabra - Vektorenzauberei

Die wohl spektakulärsten Anwendungsmöglichkeiten von Type Art liegen in der Modifikation schon fertig vorliegender Zeichensätze. Hier gibt es eine ganze Reihe von Funktionen, die mich bei der ersten Arbeit am meisten beeindruckt haben, und die Type Art sicher für jeden, der viel mit Vektorschriften arbeitet, zu einem wichtigen Wekzeug werden läßt.

Faszinierend ist die Join-Funktion. Sie erledigt das „Entgraten" solcher Schriftzeichen, die aus einzelnen Segmenten zusammengestückelt wurden. Wie Sie sicher selbst schon frustvoll erfahren haben, wird bei so gestalteten Fonts jedes einzelne Segment in einer Outline-Grafik genauso erbarmungslos gezeichnet, wie es beim Schneid-Plotter mitgeschnitten wird. Für solche Anwendungen sind diese Schriften also absolut wertlos - bis jetzt jedenfalls! Type Art berechnet bei diesen Zeichen die Schnittpunkte der sich überschneidenden Segmente und löscht in einem weiteren Schritt alle überflüssigen Linien innerhalb des Zeichens. Das Resultat ist ein sauberes Outline-Zeichen! Das hört sich vielleicht phantastisch an; im Ergebnis kann mit dieser Funktion aus allen sich überschneidenden Objekten eines Schriftzeichens oder einer importierten Vektorgrafik jeweils eine saubere Outline errechnet werden, wobei die Rechenzeit trotz der hier notwendigen umfangreichen Berechnung auf einem TT z.B. nur wenige Sekunden dauert!

Sehr schnell ist dieser Editor; kaum ein Arbeitsschritt, der nicht im Augenblick ausgeführt oder dargestellt wird. Mit der gleichen Join-Funktion lassen sich aber auch Objekte gegenseitig beschneiden, wobei ein Objekt quasi als Stanze fungiert. Noch einmal: die Rede ist immer noch von Vektorobjekten! In der 2.0-Version wird es zudem möglich sein, zusätzliche Outlines in einem frei definierbaren Abstand um oder in ein Objekt zu legen, wodurch unter anderem das Abspeichern echter Outline-Fonts in den jeweils gewünschten Stärken möglich ist. Auch für die Plott-Ausgabe bereitet es dann keine Probleme mehr, ein z.B. rotes Logo mit einer paßgenauen schwarzen Outline nacheinander auf Folie auszugeben.

Diese Funktion hat aber auch weitreichende Konsequenzen für die Font-Arbeit: unterschiedliche Schriftschnitte (light, medium, bold usw.) können dann von Type Art automatisch generiert werden! Inwieweit typografische Normen und ästhetischer Gesamteindruck erhalten bleiben, werden wir sehen. In der aktuellen Entwicklung ist dieses jedoch schon in einer eindrucksvollen Qualität möglich, wie die Beispiele zeigen.Zur „Effekthascherei" mit Vektorobjekten beinhaltet Type Art ein Rechenformular, in das beiliegende Formeln geladen oder auch neue erstellt und gespeichert werden können. Wer jetzt vielleicht an das Programm „Outline Art" der gleichen Firma denkt, denkt richtig. Nur ist die Handhabung und Ausstattung der Rechenfunktionen in Type Art um einiges besser gelöst worden; auch wenn das Handbuch in diesem Zusammenhang nur mit wenigen Beispielen aufwartet, die sich auch nicht so ohne weiteres auf andere Bereiche übertragen lassen. Objektprojektionen auf Kugel und Tonne, Drehen und Neigen sind per Knopfdruck auch ohne Formeleingabe auszuführen und werden, wenn gewünscht, gleich für den gesamten Font berechnet. Ganz bestimmt ist nicht jeder „Spiral-Kugel-Trapez-Schrift-Font" typografisch innovativ und sinnvoll; zum Erstellen von Wortmarken oder grafischen Zeichen gleich den entsprechenden kompletten Font zur Verfügung zu haben - das ist schon was.

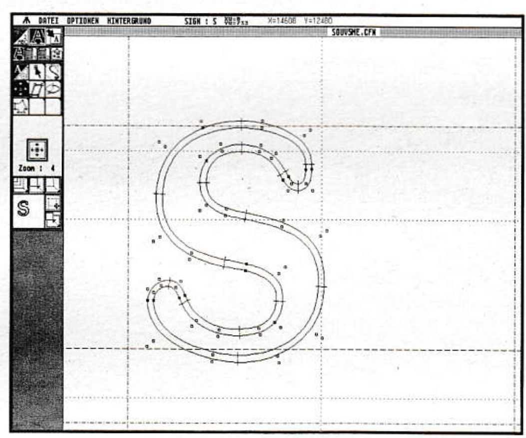


Bild 5a: Out- und Inlines generiert Type Art in der jetzt kommenden Version automatisch. In Verbindung mit den Join-Funktionen lassen sich Objekte gegenseitig ausstanzen ...

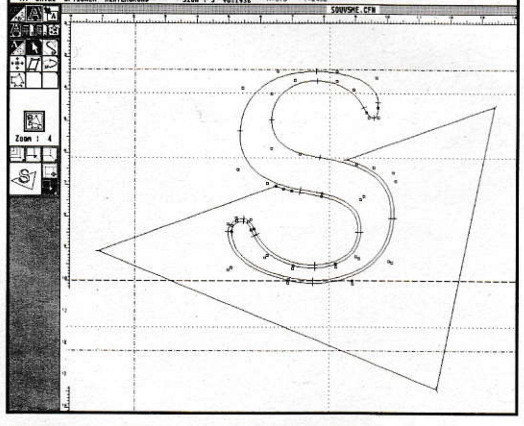


Bild 5b: ... und in wenigen Arbeitsschritten zu sauber separierten Vektorobjekten zusammenstellen. Das exakte Ineinandersetzen von z.B. mehrfarbigen Plot-Elementen ist so möglich.

Rasterbilder und Vektorisierung

Rasterbilder akzeptiert Type Art nur im IMG-Format. Wozu Rasterbilder in einem rein vektororientierten Programm? Nun, das ist der Weg, auf dem die meisten Schriften und Zeichen ins Vektorformat kommen. Mit Stift und Papier wird ein Zeichen entworfen und als Pixelbild gescannt. Für die dann folgenden Arbeiten bietet Type Art ein eigenes Bearbeitungsfeld. Hier lassen sich die Scans den unterschiedlichen Zeichenhöhen (p, k, x-Linie) genau anpassen, so daß auch unterschiedlich groß gescannte Zeichen eines Alphabets ihren genauen Platz zwischen den Ausrichtungslinien einnehmen. Eine einmal vorgenommene Skalierung kann dann für alle folgenden Zeichen übernommen werden, die Proportionen bleiben erhalten. Damit aus den so aufbereiteten Pixel-Bildern auch schöne Vektoren werden, ist zur automatischen Vektorisierung mit Linien und BÇziers das Modul „Speed Line" integriert, das auch in Calamus SL seine Dienste leistet und sehr gute Resultate bringt (ST-Computer 9/91). Auch für die Eingabe über Grafiktabletts stehen die gängigsten Treiber zur Verfügung. Nachteilig wirkt sich hier, wie in der gesamten Vektorarbeit, die interne Bearbeitungsgrenze von maximal 256 Polygonen aus, was für einen einfachen Font-Editor ausreichen mag, den viel weitergehenden Fähigkeiten von Type Art aber eine spürbare Grenze setzt. Größere Objekte werden bei Überschreiten dieser Größe nur teilweise oder gar nicht geladen. Hier wird deutlich, daß Type Art in seinen Möglichkeiten zwar auf erheblich mehr angelegt ist als ein Font-Editor; dieses „Mehr" dann aber zu bearbeiten und auszugeben, wieder auf Font-Editor-Niveau reduziert ist. Nach Auskunft der Programmautoren soll es derlei Ladebegrenzungen in Kürze denn auch nicht mehr geben.

Zur Ausgabe steht im Moment „nur" ein Plotter-Modul zur Verfügung, serienmäßig! Mit dem Erscheinungstermin dieses Artikels wird auch ein Druckformular aufzurufen sein, in dem die ganz speziellen Bedürfnisse bei der Font-Erstellung (Kontrolldrucke) berücksichtigt sind: Font-Ausdruck in Outline oder gefüllt, mit Ausrichtungslinien, Kerning-Tabellen usw. Wer einen Schneid-Plotter besitzt, kann aus Type Art heraus hier erstellte Objekte oder importierte CVGs auf Folie ausgeben. Die einzigen, aber wichtigen Einschränkungen sind die schon erwähnte Limitierung der Objektgrößen und die Darstellung der Plot-

Flächen in nur DIN-Formaten (bis A0). Für den gewerblichen Einsatz reicht das noch nicht aus. Innerhalb dieser Grenzen ist das Modul jedoch eine leistungsfähige Plotter-Software, was durch die Modifikationsmöglichkeiten in Type Art nur noch interessanter wird. Entsprechende Treiber liegen bei (z.B. für den „Camm 1"), andere können selbst editiert und abgespeichert werden.

Quo vadis

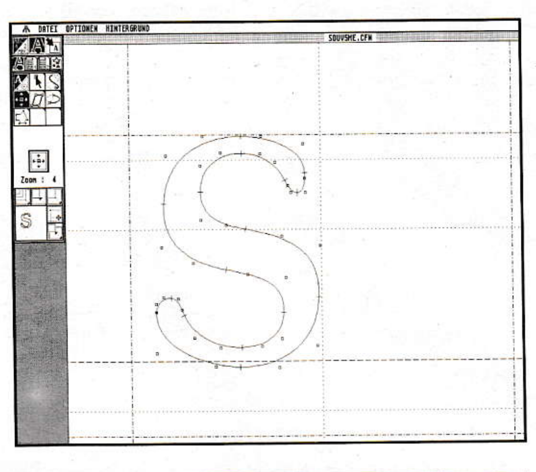
Gerade erst das Atari-Licht der Welt erblickt, und schon auf dem Weg zum Klassenbesten seiner Sparte: voll von nützlichen und sauber arbeitenden Funktionen, die es in dieser Software-Gattung zum Teil bisher noch nicht gegeben hat. Wer Vektorschriften selbst erstellt, wird an Type Art gar nicht mehr vorbeikommen. Aber auch diejenigen, die das eigene Font-Archiv um zusätzliche Fonts ergänzen oder vorhandene Schriften qualitativ verbessern oder auch überhaupt erst brauchbar machen wollen (ich denke hier z.B. an die Join- und Kerning-Funktionen), werden in Type Art ein sehr produktives und zuverlässiges Werkzeug finden.

Die weitere Entwicklung über die reinen Font-Editor-Arbeiten hinaus ist im Programm selbst ja eigentlich schon vorgezeichnet: umfangreiche Schriftbearbeitung, das Erstellen und Bearbeiten von Vektorgrafiken, Plot-Modul, demnächst gefüllte Darstellung und echte Farbverlaufsraster in definierten Zeichen (also keine Masken-Arbeiten wie in Outline Art), Text-Editor (ASCII-Import), Bearbeitung mehrerer Fenster gleichzeitig, Joblisten-Verwaltung im erweiterten Plott-Modul - ein Präzisionswerkzeug für alle vektororientierten Arbeiten tritt da in Erscheinung. Type Art einen „Font-Editor" zu nennen, halte ich darum eher für eine Untertreibung. Es gibt nicht viel Software für den professionellen DTP-Bereich, die in sich stimmig ist, innovativ dazu und rundherum funktional: Type Art gehört dazu.

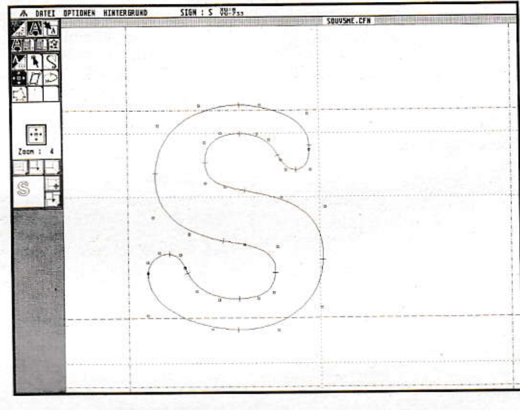
Jürgen Funcke

Bezugsadresse:

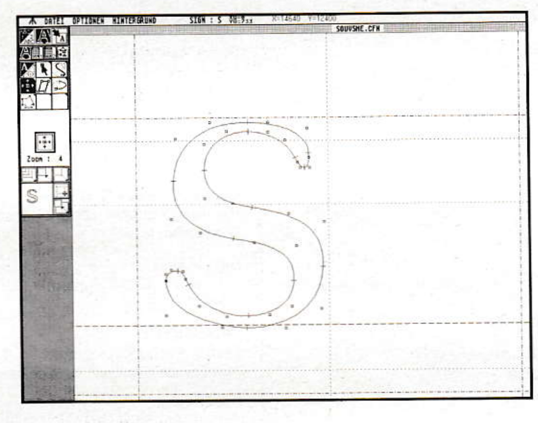
DMC
Nelkenstr. 2
W-6229 Walluf
Tel. 06123/71250



6a, b, und c: Auch das Generieren mehrer Schriftschnitte ...

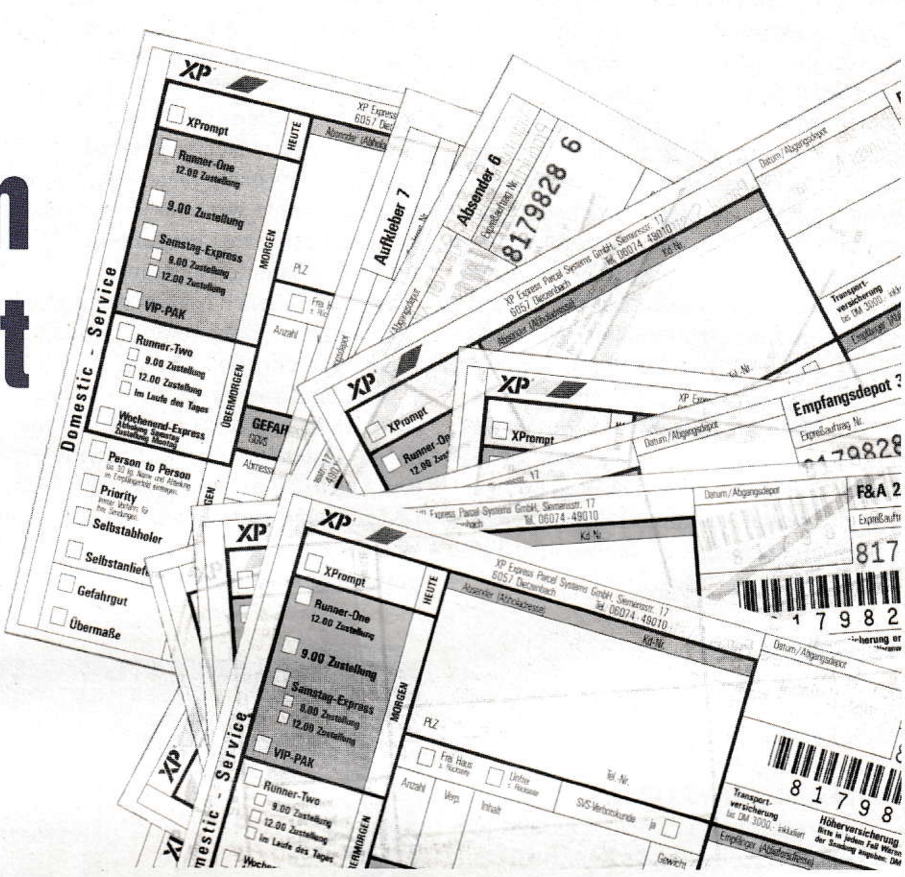


... aus einem einzigen vorhandenen Font übernimmt dann Type Art. Ein Zeichen des Ausgangs-Fonts, ...



... in unserem Beispiel die „Souvenir medium", sehen Sie in der Abbildung 6a. Andere Schnitte lassen sich nach frei einstellbaren Prozentwerten erstellen.





Formular-programme

DIPSI

Was ist die schönste Datenbank wert, wenn der Ausdruck der Daten auf Papier zu wünschen übrig läßt? DIPSI - das Kürzel eines lateinischen Mottos: describere ingeniosus pernix simplex indicii - also: gedacht zum geeigneten, schnellen, einfachen Ausdruck - hat sich zur Aufgabe gestellt, Daten aus Datenbanken auf Briefumschläge, Etiketten, Formblätter, Listen und selbst definierte Felder zu drucken.

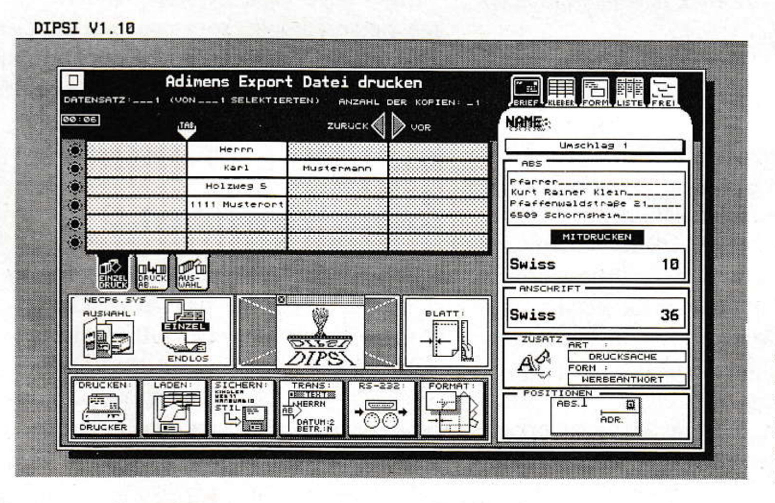
Programminstallation...

Die Installation von DIPSI (Version 1.10) übernimmt das mitgelieferte Installationsprogramm. Es fragt, ob DIPSI als Programm oder Accessory auf welchem Boot-Laufwerk mit oder ohne GDOS und mit welchem Druckertreiber installiert werden soll! Auf der Diskette befinden sich Druckertreiber und GDOS-Zeichensätze für 9- und 24-Nadel- und den Atari-804-Laserdrucker. Die automatische Installation handelt nach der in der Dialogbox angegebenen Auswahl und setzt die Prüfsumme für den Viruscheck, um eventuell später einen Virenbefall zu erkennen. Durch Umbenennung der Extension kann DIPSI.PRG auch nachträglich in ein ACC und umgekehrt gewandelt werden.

Wenn 'GDOS installieren' angegeben wurde, wird ein Auto-Ordner auf der Boot-Partition angelegt, in den das GDOS-Programm gelegt wird. Dazu schreibt DIPSI eine ASSIGN.SYS-Datei, in der der Pfad für den Druckertreiber und die Zeichensätze steht. Eine mühsame manuelle Einrichtung bleibt dem Anwender erspart. DIPSI arbeitet auch ohne den Ausgabetreiber GDOS. Briefumschläge, für die die Druckzeichen um 90 Grad gedreht werden müssen, können dann allerdings nicht mehr bedruckt werden. Für den reinen ASCII-Druck sind durch Anklicken des Drucker-Icons bei gedrückter Alternate-Taste die Anzahl der zu druckenden Zeichen und Zeilen pro Zoll sowie ein Seitenvorschub nach einer Druckseite oder am Ende des Druckvorgangs anzugeben.

Funktionen voreinstellen...

DIPSI kommt ohne Menüleiste und Drop-Down-Menüs aus. In einer Dialogbox oder über Pop-Up-Menüs sind alle Funktionen mit der Maus bzw. der Tastatur anwählbar. Dabei wurde auf kurze Mauswege geachtet. Das Hauptbetätigungsfeld besteht aus vier mal sechs Kästchen, in die



Die Arbeitsoberfläche von DIPSI präsentiert sich als Icon-Dialogbox

die Daten geladen und in denen sie beliebig positioniert werden können.

Nach dem ersten Programmstart sind einige Voreinstellungen nötig, die dann bei wiederholter Arbeit mit dem Programm gleich parat sind. Wenn mehrere Drucker gleichzeitig installiert wurden, kann im Icon 'Druckerauswahl' über ein Pop-Up-Menü der gewünschte Drucker ausgewählt werden. Ob mit Einzelblatt, Papierkassette oder Endlospapier gearbeitet wird, ist zu entscheiden. Das Icon 'Blatt' läßt die genaue Beschreibung des druckbaren Bereichs auf dem Papier zu. Die Zahlen werden im Millimetermaß eingetragen. Um diese Einstellungen zu sichern, ist das Icon 'Sichern' anzuklicken, und die Datei DIPSI.INF wird neu geschrieben. Unter 'Format' lassen sich Einstellungen für Briefumschläge, Aufkleber und Formulare angeben.

Daten laden...

Drei verschiedene Dateiarten können in DIPSI geladen werden: Eine ADIMENS-Exportdatei, BS-HANDEL- und eine NULL-Datei. Die ADIMENS-Exportdatei darf maximal achtzehn Einträge pro Datensatz (vier mal sechs Kästchen!) mit höchstens 9999 Datensätzen, sofern der Arbeitsspeicher dies zuläßt, aufweisen. Bei mehr Einträgen pro Datensatz bietet ADIMENS die Möglichkeit, eine Auswahl der Daten zu treffen, bevor die Exportdatei angelegt wird.

Eine zu ladende BS-HANDEL-Datei kann per Pop-Up-Untermenü spezifiziert werden in: Es wird gefragt, ob alle Adreßdaten, Kunden-, Lieferanten- oder Personaldaten geladen werden sollen. DIPSI sucht dann selbständig eine Datei 'Adressen.Dat' auf dem Speichermedium. Der Zugriffspfad auf die BS-HANDEL-Datei kann mit 'Sichern' festgelegt werden.

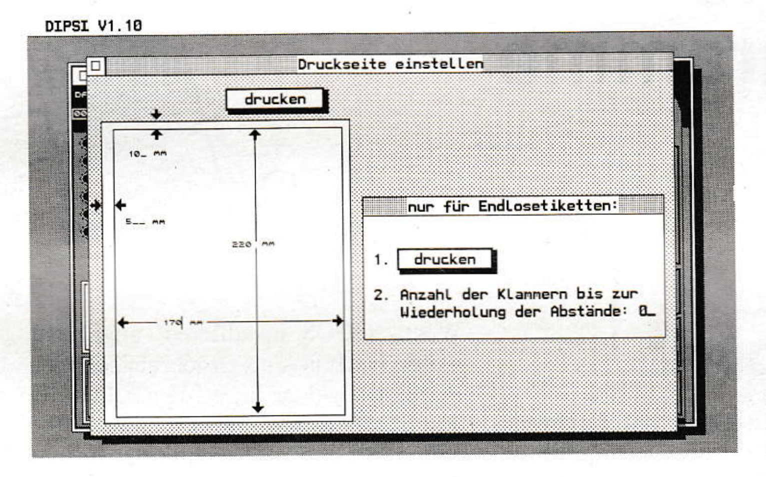
Die NULL-Datei stellt im Speicher einen Datensatz bereit, den man mit Konstanten füllen kann. Aufkleber, die immer den gleichen Aufdruck haben, werden so hergestellt.

Daten verschieben...

In den Kästchen werden die geladenen Daten satzweise plaziert. Um die Positionen der einzelnen Datenfelder zu bestimmen, klickt man ein besetztes Feld mit der Maus an, hält die linke Maustaste gedrückt und schiebt das Feld an die gewünschte Stelle. Beim Verschieben in ein bereits belegtes Feld führt die Aktion dazu, daß es zum Positionstausch zweier Felder kommt. Man kann auch ein leeres Feld per Doppelklick anspringen und Daten aus dem dann erscheinenden Pop-Up-Menü einsetzen. Zwischen den Datensätzen kann hin- und hergeblättert werden, oder man gibt die Datensatznummer zur Direktauswahl an.

Durch Doppelklick auf ein gemustertes Feld wird ein Pop-Up-Menü erzeugt, das die Auswahl zwischen einem Datenfeld und einer Konstante bietet. Bei Auswahl einer Konstante wird eine Box zum Eintragen eines Textes dargestellt, der über alle sechs Datenfelder hinweg erscheint. Konstanten können nachträglich geändert oder entfernt, Datenfelder gelöscht und wieder zurückgeholt werden.

Wird ein Datenfeld durch Doppelklick ausgewählt, erscheint ein Pop-Up-Menü, um das Datenfeld mit Schriftattributen für den Ausdruck zu schmücken, als Sortierfeld zu bestimmen oder um Trennzeichen zwischen den Datenfeldern zu setzen. Die Sortierung läuft wahlweise auf- oder abwärts und kann auch nach dem normalen Datum oder dem Geburtsdatum durchgeführt werden.



Manuelle Einstellung für eine Druckseite

Zwischend der ersten und zweiten Spalte über den Verschiebfeldern sitzt der Tabulator, der durch Anwählen das Druckergebnis so beeinflußt, daß Spalten abgesetzt aufs Papier gebracht werden können, z.B. um die Postleitzahl in einem Adressenaufdruck nach links versetzt abzubilden.

Links neben den Verschiebfeldern liegen runde Knöpfe, sogenannte 'Festzeilen-Marker'. Ein invertierter Knopf bringt eine datenlose Zeile als Leerzeile zum Ausdruck, sonst rücken die Zeilen unterhalb der Leerzeile beim Druck nach oben.

Daten auswählen...

Unterhalb der Verschiebefelder befinden sich drei Icons, die das Aussehen von Gläsern haben. Das erste Glas bietet den 'Einzeldruck' des gerade in den Verschiebefeldern zu sehenden Datensatzes an, das zweite hingegen erlaubt den 'Druck ab..' dem angezeigten Datensatz. Das dritte Glas ruft bei Anklicken eine Dialogbox auf, in der die 'Auswahl' der zu druckenden Daten getroffen werden kann. Die angezeigten Daten können im Fenster der Dialogbox vor- und zurückgeblättert werden. Mit dem Auswahl-Icon wird bestimmt, welches Datenfeld aus der Datei in dem Fenster erscheinen soll, also alle Namen, alle Geburtstage usw. Nun lassen sich zur Selektion die gewünschten Daten durch Mausklick auswählen und erscheinen invertiert.

Vielfältig ausdrucken...

Die Ausgabe der Daten kann auf fünf verschiedene Arten geschehen: als Brief, Kleber, Formular, Liste oder in frei positionierte Felder, wozu fünf umgestülpte Gläser in der Dialogbox rechts oben dienen.

Soll ein Briefumschlag bedruckt werden, stehen vier Zeilen zur Verfügung, um den eigenen Absender frei einzutippen. Unter dem Absenderfeld erscheint der Zeichensatzname für den Absenderausdruck. Bei Anklicken erscheint eine Box, die alle zur Verfügung stehenden Zeichensätze zeigt. Geht man mit der Maus auf den Pfeil des Eintrags, zeigt ein Untermenü die verschiedenen Druckgrößen an. Die gewünschte Größe wird durch Mausklick gewählt. Für die Adresse wird der Zeichensatz gesondert, aber nach dem gleichen Verfahren ausgewählt. Briefumschläge können nur mit dem GDOS-Ausgabetreiber gedruckt werden. Auf dem Briefausdruck kann die Angabe der Versandart und -form mitgedruckt werden. Zwei unterschiedliche Pop-Up-Menüs bieten eine große Auswahl an Einträgen dazu an. Die Textattribute für diese Zusatztexte lassen sich individuell angeben. Die Druckpositionen der Daten auf dem Briefumschlag mißt man mit einem Lineal auf dem Umschlag aus und trägt die Ergebnisse in die Umschlagmaßfelder ein, die nach Anwählen des Umschlag-Icons erscheinen. Sechs

Raubkopien

lohnen sich nicht mehr...

Action / Simulationen..

Bloodwich	39,-	Bio Challange	19,-
Colorado	29,-	Cloud Kingdom	29,-
The Krystal	49,-	F-16 Falcon	59,-
F-16 Mission 1	59,-	F-19	79,-
Lombard RAC	39,-	Lemmings	69,-
F 15 Str.Eagle2	99,-	Speedball	19,-
Pirates	59,-	Turrican	29,-
Their Finest H.	79,-	Turrican 2	49,-
Team Suzuki	69,-	XENON 2	29,-
Railroad Tycon	89,-	Elvira	69,-
Battle of Brit.	79,-	Ferrari F1	39,-
Stargilder II	29,-	PacMania	29,-
Ninja Spirit	39,-	Ninja Warrior	29,-
Flight Of The Intruder			79,-

SAMPLER ...

Action ST (Masters of Universe,3D Galax, Trailblazer,Deflector,Northstar) 29,-
Hit Disk 1 (Goldrunner,Jupiter Probe, Slaygon, Karate Kid II) 29,-
Premier C.1 (Nebulus,Exolon,Zynaps, Netherworld) 29,-
Heroes (Licence to kill,BarbarianII, Running Man,Star Wars) 39,-
Triad Vol 2 (Menace,Tetris,Baal) 29,-

adventures ...

SIERRA je 59,-

Kings Quest 1, 2, 3, 4, Space Quest 1, 2, Codename Iceman,Manhunter San Franc. Conquest Of Camelot, Mother Goose.

Gold Rush	49,-	Space Quest 3	89,-
Larry 3	69,-	Black Cauldron	49,-

Infocom ab 29,-

Ballihoo	49,-	Beyond Zork	49,-
Enchanter	49,-	Hitchhiker G.	49,-
Hollywood Hijinx	39,-	Leather God.	49,-
Lurking Horror	49,-	Moonmist	49,-
Seastalker	29,-	Plundered H.	49,-
Spellbreaker	39,-	Starcross	79,-
Stationfall	49,-	Suspect	39,-
Deadline	49,-	Trinity	49,-
Wishbringer	39,-	Witness	39,-
Zork I/II/III	49,-	Infidel	59,-

Infocom InvisiClues Lösungshilfen je 19,-

Liste gratis. Computertyp angeben!
Versandk: Vorkasse 5,-; Nachnahme 7,-
Softwareversand U. Wandrer Postfach 4
W- 3067 Lindhorst ☎ 05725/5426

Bossart - Soft presents

Games

Suchen Sie preiswerte Original Atari Programme? Wir haben Sie. Verlangen Sie unsere Liste.

Sonstiges

Haben Sie irgend ein Programm in der Schweiz nicht bekommen ? Nervt Sie ein Computervirus ? JA, dann rufen Sie uns an.

7.12.1991

Grosse Computer Börse in Emmenbrücke. Das Datum das Sie sich merken sollten.

BOSSART - SOFT
P.B. 5146
CH-6020-Emmenbrücke-3
041/45'82'84

MAXIDAT

- die umfangreiche Datenbank - für alle Atari ST / STe / TT



Benötigen Sie eine relationale Datenbank, die etwas mit den Daten anfängt? MAXIDAT kann viel. Hier das Wichtigste im Überblick:

- **Integrierter**, einfach zu bedienender **Texteditor.**
- **Serienbriefe** in Verbindung mit dem integrierten Texteditor oder auch einem beliebigen anderen (z.B.: Tempus, 1st-Word, EdiMax, Thats Write, Edison).
- **Rechnen innerhalb Datenfeldern** (Feldinhalte, Klammern, + - * /).
- **Summenbildung bei Listendruck.**
- **Diagrammerstellung** (Linien-, Balken- und Tortendiagramme), z.B. Erstellung von Aktiencharts.
- **Zugriff auf externe Textdatei.**
- **Bildverarbeitung:** Je Datensatz Zugriff auf externes Grafikbild (Formate: Doodle, Stad, Neochrome, Degas). Automatische Auflösungsanpassung.
- 'Diashow' für Werbezwecke und einfache Trickfilme **mit raffiniertem Bildaufbau.**
- **Selektionsmöglichkeit** zur Beschränkung der Datensatzausgabe ('Filter').
- Beliebige Datenbestände miteinander verknüpfbar und durch Selektion frei trennbar.
- **Ermittlung von Min, Max, Summe, sowie Durchschnitt** aller Datenfeldreihen für Bilanzen.
- **Drucken in allen Variationen und Formen** (Etiketten, Formulare, Listen, Rechnungen, Mahnbriefe...) mit umfangreichen Möglichkeiten (Seitennummer, Spaltensatz, Datum, Kopf- und Endetext,...)
- **Alle Drucker** (auch HP- und Atari- Laser) **werden unterstützt,** wobei der Druckertreiber ggf. selbst im Programm angepasst werden kann.
- **Listenausgabe auf Monitor, Drucker und Datei.**
- **Suchen** nach allen Feldern sowie global und in externen Texten.
- **Sortieren** nach allen Feldern **mit zweifacher Untersortierung** (z.B. Name, Ort, Geburtsdatum).
- **Fünf Feldtypen:** Text, Zahl, Datum, externer Text und externe Grafik plus externes Programm
- Je Datei **relationaler Zugriff** auf eine weitere Datei (z.B. Kunden / Bestellungen).
- **Leistungsfähiger Editor** zur Beschriftung der Datensätze (mit Datum, Undo, Reg-Nr, Sonderzeichentabelle, Zeilenpuffer, Floskeltasten, Help,...).
- **Komplette Datensätze kopierbar** (Copy/Paste).
- Zehn Marken zum Anspringen von Datensätzen.
- **Programmaufruf ohne MAXIDAT zu verlassen.**
- Auf Wunsch **verschlüsselte Speicherung** der Datenbestände **mit Passwortschutz.**
- **Einzelne Datenfelder ausblendbar.**
- Zahl der Datensätze je Datei nur vom Speicher abhängig (Mega ST4: max. 100 000 Stück).
- **Dynamische Datenstruktur,** daher optimale Speicherausnutzung (keine Füllzeichen).
- Besonderer Wert wurde bei der Programmerstellung auf eine **einfache Bedienung** und **hohe Arbeitsgeschwindigkeit** gelegt ('C', 'Assembler').
- **Datenübernahmemöglichkeit** aus zahlreichen anderen Programmen (z.B. 1stAddress, Superbase, Datamat), sowie **Export in fast allen Dateiformaten** zum Zwecke des Datenaustausches.
- MAXIDAT ist ein **nicht kopiergeschütztes,** eigenständiges Programm und nicht etwa ein Accessory.
- MAXIDAT wird bereits seit **Jahren von zahlreichen Anwendern im privaten und geschäftlichen Bereich** eingesetzt. Auch wir verwalten unsere Kunden ausschließlich mit MAXIDAT. Somit ist unsere Datenbank **frei von 'Kinderkrankheiten'** und hat sich im harten Alltagseinsatz bewährt.
- Umfangreiches deutsches Handbuch sowie **Hotline mit dem Autor** im Preis inbegriffen.
- **Günstiger Upgrade-Service.**
- MAXIDAT wurde in 'PD-Journal 5/90', 'TOS 7/90' und 'ST-Computer 3/91' getestet. Testberichte gratis.

MAXIDAT ist eine der umfangreichsten Datenbanken für den Atari ST. Überzeugen Sie sich durch die Testversion. Haben Sie weitere Fragen? - Schreiben Sie uns. Übrigens sind wir schnell. Ihre Anfragen werden innerhalb von sechs Stunden bearbeitet.

MAXIDAT kostet DM 87.-

Die Testversion gibt's für DM 10.- inkl. Versand (Vorkasse)
Versandkosten: Vorkasse DM 3.00, NN DM 5.00, Ausland DM 6.70 (nur Vorkasse)

Softwarehaus Alexander Heinrich
Postfach 1411
W-6750 Kaiserslautern
Tel. 0631-29101

T.U.M.

Soft & Hard Handels GbR Helfers Jeddeloh

ATARI® Fachhandel
Hauptstr. 67/Pf. 1105
2905 Edewecht
☎ (04405) 6809

ATARI-Public-Domain
Preise: nur 4 - 5 DM
Der Katalog
über 200 S. mit (fast) allen Serien, alphabet. Index, akt. Angebote: nur 5 DM.

..Software

Calamus V1.09N	398,-
ArtWorks	398,-
Kobold	79,-
OXYD2 + Buch	60,-
Spacola + Buch	60,-

..HD KIT's
für MEGA STE/TT

48 MB Seagate	498,-
105 MB Quantum	1129,-

..mehr RAM STE

auf 2 MB	219,-
auf 4 MB	435,-

..Mäuse

That's a Mouse	79,-
Logimouse	85,-

..Disketten
TDK MF2DD Bulkware
(ohne Label, ohne Shutterdruck)
50 St. 60,- 100 St. 115,-

..MEGA STE PAKET
Mega STE 2, SM124, 48 MB HD 24Nadler KXP1123+Kabel, 20 Disk, Mausmatte, Einsteiger Buch, Einsteiger PD-Paket
unser Preis 3.333,-

Versand erfolgt durch DBP als Brief bzw. Wertpaket zzgl. Versandkosten.



Font - Collection

neue Fonts für Ihren Calamus®
®Calamus ist eingetragenes Warenzeichen der Fa. DMC GmbH, 6229 Walluf

Americano	Impuls
Alt berlin reg.	Elan light
Comic Strip	Florence
CARDPLAY	KINSLEY
Floating light	Octave
PAINTCUT	Smallface light
Metro light	☎→⇩□✂

NEU DTP-Vektor Grafiken



Info anfordern bei:



Fontware
Hauptstr. 67/Pf. 1105
D-2905 Edewecht
☎ 04405/6809 Fax: 228

Preisknüller

10 PD-Disketten aus ST-Computer oder unserer Katalogdisk nur 29,-

< 10 Disks	nur 3,50/Disk
ab 10 Disks	nur 2,90/Disk
ab 20 Disks	nur 2,45/Disk
ab 50 Disks	nur 1,99/Disk

Katalogdisk mit allen Diskinhalten einschl. Versand gratis!

Sehr gute PD-Pakete je 10 Disks nur 25,-

PD-Set B	(Spiele s/w)
PD-Set C	(Spiele Farbe)
PD-Set E	(Utilities)
PD-Set F	(Grafik+Pics)
PD-Set G	(Midi+Musik)
PD-Set I	(Signum-Fonts)
PD-Set K	(Erotik s/w)
PD-Set L	(Erotik Farbe)

TeX-Komplettset V2.0 (11 Disks) .... nur 29,-
Gnu C++ (5 Disks) ..... nur 15,-

SW-Haushalt ist eine sehr einfach zu bedienende Haushaltsbuchführung mit vielen Auswertmöglichkeiten und Handbuch. Auch für die Schweiz und Österreich geeignet.
Preis ....nur **49,-**

● 24-Nadeldrucker ●

STAR LC 24-10	599,-
STAR LC 24-200	798,-
Epson LQ 400	549,-
Epson LQ 550	798,-
Citizen Swift 24	749,-
Fujitsu DI 1100	949,-
Fujitsu DL 900	749,-
NEC P20	749,-
NEC P30	998,-
NEC P60	1298,-
NEC P70	1659,-

● Tintenstrahldrucker ●
HP-DeskJet 500 1099,-

●●● Laserdrucker ●●●

Topangebot:
Laserdrucker Canon LBP-4 (512 KB) ... nur 1599,-

●●● Festplatten ●●●

Quantum Einbaufestplatten für MEGA ST-Serie.

52 MB/17 ms	749,-
105 MB/17 ms	998,-

Externe Festplatten von Quantum mit SCSI-Bus.

52 MB/17 ms	989,-
105 MB/17 ms	1248,-
210 MB/15 ms	1998,-
425 MB/14 ms	3898,-

Speichererweiterungen

2 MB für 1040 STE	198,-
2/4 MB alle ST's	348,-
4 MB alle ST's	575,-

Auf alle Festplatten 2 Jahre Garantie.

●●● Monitore ●●●

ATARI SC 1435	549,-
ATARI SC 1224	649,-
NEC Multisync 3D	1249,-

Versandkosten für Software:
VK 5,-, NN 7,- / Ausland nur VK 10,-
Versandkosten für Hardware:
Nur per NN plus 15,-

SW-SOFTWARE
Soft- und Hardwarevertrieb
Beethovenstr. 10, 7938 Oberdischingen
Tel. 07305/8325

verschiedene Briefumschlagsarten mit entsprechenden Einstellungen, die bei erneutem Programmaufruf automatisch präsent sind, können jeweils unter einem Namen abgespeichert werden.

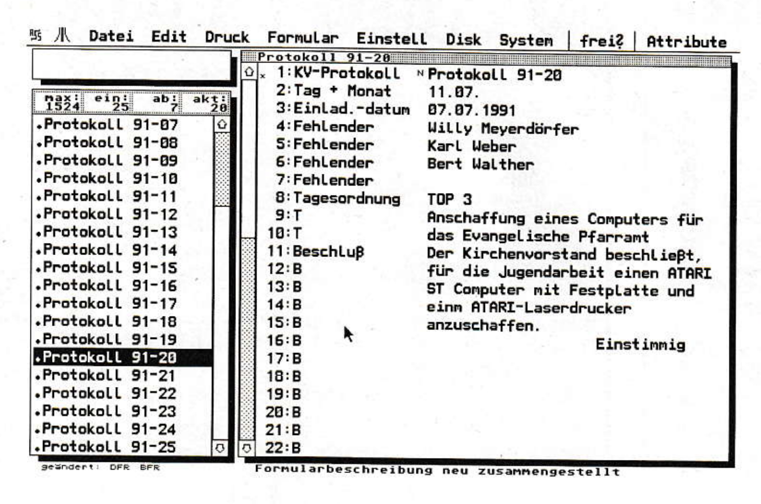
Sollen Daten in ein Formular, beispielsweise der Name in eine Urkunde, eingetragen werden, müssen zunächst der obere und untere Druckrand sowie die Ausmaße des horizontalen und vertikalen Druckfeldes in Millimeterangaben in eine Blattvorlage auf dem Bildschirm geschrieben werden.

Wird das kopfstehende Glas 'Kleber' angewählt, ändert sich die Darstellung der Dialogbox im rechten Teil. Ein Etiketten-Auswahlfeld wird dargestellt, neben dem 'Alle' und 'Keiner' steht. Wird der erste Button angeklickt, heißt das, alle Etiketten werden ausgedruckt. 'Keiner' invertiert alle Felder und gibt die Möglichkeit, einzelne bestimmte zum Ausdruck auszuwählen. Diese manuelle Auswahl ist nur mit GDOS-Ausgabetreiber funktionsfähig. In der Box 'Form' kann zwischen Etiketten mit und ohne Traktor gewählt werden, wobei jeweils ein Untermenü eine große Anzahl vordefinierte Etikettengrößen in DIN-Norm zur Auswahl stellt.

Für die Etiketten können postalische Zusatzangaben wie bei Briefumschlägen gewählt werden. Ebenso sind bis zu sechs verschiedene Einstellungen unter einem Namen abzulegen.

Beim Listendruck erhält das Verschiebefeld in der ersten Zeile durchgehend ein Konstantenfeld für eine Titelzeile, die ausgedruckt wird, wenn der runde Knopf links daneben selektiert wurde. Alle Felder darunter sind spaltenorientiert. Spalte eins erhält dann in den verbleibenden fünf Feldern z.B. das Datum aus fünf verschiedenen Datensätzen, Spalte zwei den Namen usw. Eine Überschrift kann der Liste im Einstellungsfeld zugewiesen werden. Ein linker Rand von 2,5 cm, die Seitennummer, das aktuelle Datum und eine Datensatznumerierung können getrennt ein- und ausgeschaltet sein. Das Aussehen der Liste läßt sich verschieden angeben: Der Titel, jeder Datensatz oder jeder fünfte bzw. zehnte Datensatz kann unterstrichen gedruckt werden. Vertikale Striche oder eine Einrahmung der gesamten Liste sind möglich. Die Strichdicke ist für jedes Feld variabel.

Das letzte gestürzte Glas namens 'Frei' bietet einen Ausdruck der Felder an Positionen, die für jedes Feld in x- und y-Koordinate eingegeben werden. Der zu druckende Text kann im Hoch- oder Querformat zu Papier gebracht werden. Auch hier sind wieder bis zu sechs Einstellungen zu sichern.



Der Arbeitsbildschirm zeigt links das Namens- und rechts das Arbeitsfenster von FORMULARplus.

Gekonnt transferieren...

DIPSI gibt die Daten mit und ohne GDOS-Ausgabetreiber über den Drucker aus. Im Accessory-Betrieb ist ein Transfer zu anderen Programmen möglich. Wird das Icon 'Trans' angeklickt, werden die Daten zwischengespeichert und nach Programmende in den Tastaturpuffer geschrieben. Wurde DIPSI als ACC aus einem Textverarbeitungsprogramm aufgerufen, bedeutet das, die Daten werden nun wie von Geisterhand geschrieben. Die gedrückte Alternate-Taste unterbricht diesen Vorgang.
Die Ausgabe kann des weiteren über die serielle und parallele Schnittstelle und den Midiport erfolgen. Bei gedrückter Alternate-Taste kann die Einstellung im 'RS-232'-Icon getroffen werden. Die Baud-Rate ist in Vorgaben von 50 bis 19200 einstellbar.

Fazit

DIPSI läuft auf dem ST mit mindestens 1 MB Speicher und auf dem TT in monochromer Auflösung. Auch mit einem Großbildschirm hat das Programm keine Probleme. Das sehr schöne Handbuch - kartonierte Blätter im gepflegten Ringbuch - führt auf 102 Seiten sauber und unkompliziert in die Bedienung des Programmes ein.
Für den Preis von 128,- DM erhält man ein Programm, das Daten aus ADIMENS und BS-HANDEL einliest und vielfältig gekonnt zu Papier bringt. Als wünschenswert empfinde ich, daß DIPSI in die Lage versetzt wird, Daten weiterer Datenbanken bzw. Programme einzulesen und auszudrucken.

Bezugsadresse:

ICP-Verlag Leserservice
Innere-Cramer-Klett-Straße 16
8500 Nürnberg

FORMULARplus

Wer mit einer Vielzahl von gespeicherten Daten umgeht und diese hin und wieder auch in Formularen zu Papier bringen muß oder will, wird sich schon lange nach einem Programm gesehnt haben, das beides miteinander in geschickter Weise verbindet. Vor allen Dingen gewinnt ein solches Programm an Wert, wenn sich die gleichen Daten ohne große Anstrengung in ganz unterschiedliche Formulare stecken lassen , um sich die Arbeit so einfach wie möglich zu machen.

Anspruch

FORMULARplus (Version 3.01) erhebt den Anspruch einer Datenbank mit paßgenauem Positionsdruck und ist eine umfangreiche Weiterentwicklung früherer Programmversionen. 1 MB Arbeitsspeicher ist das Minimum, um mit FORMULARplus sinnvoll arbeiten zu können, mehr ist in jedem Falle ratsam. Die Programmsteuerung erfolgt über Drop-Down-Menüs oder über die Tastatur. Der Arbeitsbildschirm, der nach dem Programmstart zu sehen ist, teilt sich in Hinweis-, Namens- und Anzeigefenster. Das Hinweisfenster gibt Aktivitätsmeldungen während des Programmlaufs aus. Im Namensfenster erscheinen die Datensätze nach dem Identifikationsfeld im Formular aufgelistet. Das Anzeigefenster gibt die Feldbezeichnungen des Formulars mit den jeweils eingegebenen Daten eines Datensatzes an. Die beiden letztgenannten Fenster sind vertikal scrollfähig, wenn die Auflistung den Bildschirmbereich überschreitet.

Um es gleich vorwegzunehmen: Das Programm speichert Formular- (*.BFR) und Datendatei (*.DFR) getrennt ab. Der große Vorteil liegt u. a. darin, daß sich zu einer Datendatei mehrere Formulardateien gesellen können. Das erspart die erneute Eingabe derselben Daten für ein anderes

Formular! Beim Erstspeichern ist darauf zu achten, daß jedes Formular und jede Datendatei eine Kennungszahl erhält. Beide Dateien können wiederum nur zusammen geladen werden, wenn sie die gleiche Kennung haben. Dasselbe gilt für die angelegten Felder, wo sie für zusammengehörige Formular- und Datendateien in der Anzahl übereinstimmend sein muß. Mit der Formulardatei werden mehr als zwanzig nötige Einstellungen abgespeichert.

Für Formular-, Daten-, Makro-, und Druckerdatei ist nach dem allerersten Programmstart der Pfad einzustellen. Eine sich beim Programmstart selbstladende Datei ist auf Wunsch einstellbar. Zu jeder Datendatei kann eine eigene Parameterdatei angelegt werden, die wichtige Einstellungen enthält und beim Laden einer Datei mitgeladen wird. FORMULARplus ist außerdem in der Lage, bis zu zehn Makrodateien automatisch mit einer Datei zu laden. Jede Makrodatei darf dabei bis zu 30 KB groß sein.

Der Kreativität ist keine Grenze gesetzt

Ob Sie eine Banküberweisung ausfüllen oder ständig Rechnungen schreiben, ob Sie Ahnentafeln erstellen oder Zeugnisse zu schreiben haben, ob Sie Serienbriefe verfassen oder Etiketten brauchen - kein Problem. Wenn einmal die Hürde der Formularerstellung genommen ist, bleibt nur noch die Ein- und Ausgabe der Daten. Änderungen am Formular können selbstverständlich jederzeit in vielfältiger Hinsicht vorgenommen werden.

Ein Formluar erstellen heißt, die benötigten Datenfelder mit maximaler Eingabelänge zu benennen und horizontal und vertikal in Millimeterangaben mit einer Nachkommastelle zu positionieren. Dazu ist es nötig, ein Lineal zur Hand zu nehmen und den zu jedem druckfähigen Datenfeld gehörigen x- und y-Anfangspunkt auszumessen. Der Übersichtlichkeit wegen ist es sinnvoll, über die druckbaren Datenfelder hinaus Benennungsfelder einzugeben, die auch der Zählung und Sortierung des Datensatzes dienen helfen.

Alles eine Frage des Stils

Eine Stärke von FORMULARplus: Jedem Datenfeld lassen sich Attribute zuordnen, die Einfluß bei der Anzeige oder beim Ausdrucken auf das entsprechende Feld nehmen. Die Formularfelder, die später ausgedruckt werden sollen, müssen das Druckattribut erhalten. Da jedes Formular maximal neun Seiten haben darf, kann hier gleich die Seite des Drucks mit eingegeben werden. Verschiedene Schriftarten, die sich mischen lassen, sowie zentrierter, rechtsbündiger und Blocksatz-Ausdruck geben dem Formular seinen Stil. Ein Datenfeld ist als Identifikationsfeld zum Erkennen des Datensatzes zu vergeben. Zwei Sortierfeldsymbole können vergeben werden.

Wer ein Adreßformular anlegt und darin Vorname und Name bzw. Postleitzahl und Wohnort in getrennten Feldern eingibt, aber nachher hintereinander ausdrucken will, kann das mit dem Attribut 'Felder vereinigen' erreichen; die Eingabe eines Leerzeichens hinter Vorname bzw. Postleitzahl aber ist nicht zu vergessen. Ein Attribut zur Auffüllung der maximalen Eingabelänge von Feldern, beispielsweise in einer Tabellenliste für Beträge, steht zur Verfügung. Aufgefüllt wird wunschgemäß von links oder von links und rechts gleichzeitig.

Eine sehr schöne Möglichkeit besteht darin, das aktuelle Datum in siebzehn(!) verschiedenen Formaten festzulegen. Beispiele: Bei amtlichen Vordrucken kommt es vor, daß die Jahreszahl schon eingedruckt ist. Dann genügen Tag und Monat als freie Eingaben. FORMULARplus ist auch in der Lage, zu dem aktuellen Datum eine vorher einzugebende Anzahl von Tagen hinzuzurechnen, was für eine 'Zahlbar bis'-Rechnung praktisch ist. Das aktuelle Datum zum Zeitpunkt des Ausdrucks wird auf Wunsch automatisch eingesetzt.

Weitere Feldattribute wandeln Eingaben in Großbuchstaben oder Zahlen in Wörter, führen zur Übernahme von Daten aus anderen Feldern des gleichen Datensatzes oder übernehmen Vorgabedaten, beides, ohne zusätzlichen Speicherplatz zu beanspruchen. Gesperrte Felder übernehmen nur Vorgabedaten und werden bei der Dateneingabe übersprungen. Um nur bestimmte Daten eines Formulars als ASCII-Text ausfilternd zu speichern, können diese markiert werden. Bedingungsfelder lassen die Ausgabe der Vorgabedaten zunächst einmal offen. Als Zahlenfeld definierte Felder ermöglichen nur die Eingabe von Zahlen mit und ohne Nackkommastellenvorgabe.

Ein kleiner Texteditor wird mit dem Attribut 'Blockzeile' für mehrere aufeinanderfolgende Eingabefelder installiert. Die als Block definierten Felder betrachtet FORMULARplus als zusammengehörig. Ein Absatzende ist markierbar, manuelle Worttrennung möglich, Zeilen lassen sich hochholen und anhängen. Zeilen, die länger als die maximale Eingabelänge sind, werden optisch hervorgehoben. Am Ende der Eingabe des Fließtextes geht man in die erste Zeile und löst mit 'Shift-Return' einen automatischen Zeilenumbruch aus.

Auch Bilder verwaltet FORMULARplus. Automatisch werden folgende Bildformate erkannt: Screenformat (32KB für ST, 153600 Bytes für TT), Degas (32034 Bytes), STAD und GEM-Image-Bilder. Ein als Grafikfeld apostrophiertes Feld speichert Laufwerk, Pfad und Größe der Bilddatei, ohne das Bild selbst zu übernehmen, was Speicherplatz spart. Zwar kann ein Bild in FORMULARplus angezeigt, aber nicht ausgedruckt werden.

Im Angebot: Erleichterungen

Die Möglichkeit, 'Vorgabedaten' in Formulare einzutragen, erspart stupides Neutippen der ewig gleichen Daten in neuen Datensätzen. Daten, die also für alle Datensätze Gültigkeit besitzen sollen, werden hier benannt. Felder, die das Attribut 'P' besitzen, holen sich ihre Daten (z.B. einen Briefkopf) erst beim Ausdruck aus den Vorgabedaten und sparen so unnötigen Speicherplatz. Gibt man für das Datum das entsprechende Symbol ein, wird dieses beim Dateneingeben in das Tagesdatum gewandelt. Genauso kann für die Eingabe der Tageszeit, mit und ohne Zusatztext 'Uhr', verfahren werden. Das Symbol für die Datensatznumerierung erzeugt die fortlaufende Datensatznumer. Trotz Vorgaben kann ein Datensatz auch individuell geändert werden. Im Nachhinein sind für alle Datensätze in differenzierter Weise Vorgaben einsetzbar.

Datenfelder zum Rechnen mit den vier Rechenarten zu veranlassen, ist ohne weiteres durchführbar. Zur Definition der Felder als Rechenfelder erscheinen alle Felder auf der linken Bildschirmseite und rechts die Rechnerfunktionen. In der Mitte werden die Rechenformeln eingegeben. Konstante Zahlen für bestimmte Felder können direkt eingetragen werden. Jede Rechenformel kann auch Ergebnisse, die von einer anderen Formel errechnet wurden, übernehmen. Das Programm rechnet mit bis zu sieben Nachkommastellen und Zahlen mit höchstens dreizehn Stellen. Die Funktion 'Rechenformeln-Spezial' gestattet sogar nachträgliche Rechenoperationen. Nachdem die Reihenfolge der Eingabefelder in der Formulardatei festgelegt ist, kann zum Komfort der späteren Dateneingabe jederzeit die Eingabereihenfolge geändert werden. Gesperrte oder vorgabedatenbelegte Felder schiebt man an

den Schluß des Formulars, um bei der Dateneingabe durch so wenig Scroll-Arbeit wie nötig Zeit zu gewinnen. Mit der Änderung der Formularfeldreihenfolge werden auch die sich im Speicher befindlichen dazugehörigen Daten automatisch umgeschaufelt.

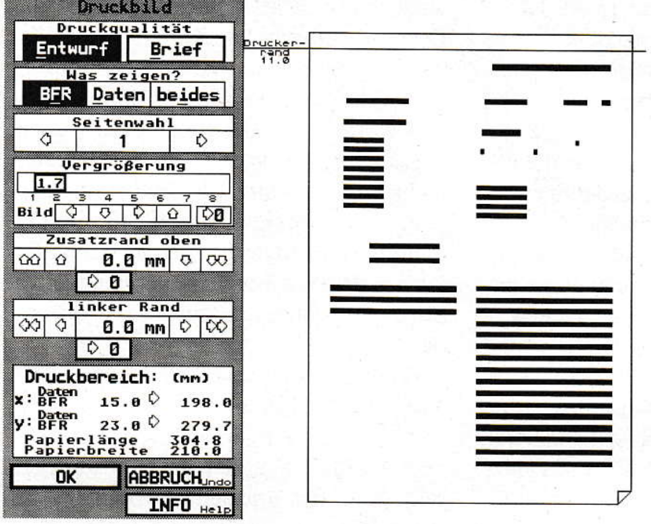
Die Übersicht behalten

Nachdem ein Formular erstellt ist, läßt sich unter dem Menüpunkt 'Übersicht' ein optischer Blick auf die als Linie dargestellten Druckpositionen und Drucklängen der Felder im linken Teil des Fensters werfen. Ein gestrichelter Rahmen gibt die Papiergröße an. Rechts daneben stehen tabellarisch alle wichtigen Informationen des erstellten Formulars. Soll ein Feld oder ein ganzer Seitenausschnitt verschoben werden, geht das mit einer Umrahmungslinie, die durch die festgehaltene linke Maustaste erzeugt wird. Die neuen Feldpositionen werden daraufhin vom Programm errechnet. Eine Verschiebungsgenauigkeit von 0,1 Millimeter ist erreichbar.
Im Tabellenmenü, in das man vom Arbeitsbildschirm aus mit 'Control'- Mausklick auf eine Zeile im Anzeigefenster gelangt, werden die Datenfelder nebeneinander angezeigt. Ganz links steht hier immer das Idenfikationsfeld. Daneben folgen die Felder ab dem auf dem Arbeitsbildschirm angeklickten. So verschafft man sich einen raschen Überblick und kann die Tabelle auch noch ausdrucken. Die Feldbreiten in dieser Tabellenanzeige sind variabel.

Die Eingabe vor der Ausgabe

Das Eingeben der Daten geschieht in einem Fenster, das links die Feldbezeichnungen und daneben die Leerstellen anzeigt, die aufgefüllt werden können. Vor dem jeweiligen Eingabefeld stehen Attributszeichen, die Feldbesonderheiten anzeigen. Nach Abschluß der Eingabe kehrt das Programm über 'Control-Return' zum Arbeitsbildschirm zurück. Datensätze können geändert, vervielfacht (wichtig für Serienbriefe) gelöscht, sortiert, und verglichen werden. Eine umfangreiche Suchen-Ersetzen-Funktion steht zur Verfügung.

Beim Speichern der Datensätze kann ein Paßwort eingegeben werden, das beim Laden zum Schutze der Daten vor falschen Benutzern dann abgefragt wird. Zusätzlich werden die Daten verschlüsselt abgespeichert.



Ein Preview-Bildschirm zeigt rechts das spätere Druckbild an. Im linken Bereich werden Einstellungen vorgenommen.

Der Ausdruck macht Eindruck

Eine Druckersteuerung für verschiedene Schriftarten, Sonderzeichen und Tastaturbelegung ist im Programm integriert und kann über Dialogboxen angepaßt und verändert werden. FORMULARplus liefert zahlreiche Druckeranpassungen für Nadler und Laser mit.

In der Druckeranpassung kann ein Korrekturfaktor für den Blatteinzug angegeben werden, der bei dickeren Blättern nötig wird. Ein oberer Papierrand, der unbedruckt bleibt, kann beliebig angegeben werden.

Im Menü 'Druck' unter 'Druckbild' erscheint eine maßstabsgetreue Vorschau auf den zu erwartenden Ausdruck. Die Vergrößerung des Vorschau-Ausdrucks auf dem Monitor ist stufenlos einstellbar. Die eingestellte Papiergröße erscheint als Rahmen und der obere Druckerrand als Linie. In drei verschiedenen Modi kann diese Vorschau erscheinen: Entweder werden durch Linien die druckfähigen Datenfelder in ihrer maximalen Eingabelänge gezeigt, oder es werden nur die auszudrukkenden Texte des aktuellen Datensatzes unter Beachtung der Druckattribute, oder beides angezeigt. Von hier aus sind ein oberer und linker Zusatzrand definierbar, die bei Formularen Sinn machen, die hie und da in ihrem Vordruck voneinander abweichen. Ob der Ausdruck im Draft-Modus oder in Briefqualität gewünscht wird, stellt man hier ein.

Spezielle Druckaufträge bietet 'Druck - Spezial' an: Steuerzeichen senden, Kopf- und Fußzeilen ausdrucken, nur bestimmte Seiten mehrfach oder mehrere Datensätze satzweise oder seitenweise drucken, Spaltensatz in beliebiger Spaltenzahl und Spaltenbreite werden zur Leichtigkeit.

Helferlein willkommen

Einige Programme begleiten FORMULARplus: PRINT-FORMULAR zum Ausdruck der mitgelieferten Formularbeschreibungen, die aus FORMULARplus heraus als Info-Texte abrufbar sind. Mit PROTECT-FORMULAR kann eine Daten- oder Formulardatei einen Bearbeitungsschutz gegen versehentliches Ändern oder Löschen erhalten und die Dateikennungszahl verändert werden. MODUL-FORMULAR erlaubt, aus bestehenden Formular- oder Makrodateien Teile herauszuschneiden und zu neuen Beschreibungen zusammenzusetzen. DDL->DFR überträgt Daten vom Programm DATADISK desselben Autors nach FORMULARplus. KONVERT 2->3 konvertiert Dateien der Vorgängerversionen 2.00-2.55 ins neue Format. Mit Hilfe von MEM_ACC ist der Speicherplatz für Accessories in FORMULARplus zu reservieren. MAKE_MFR erzeugt aus Makrodateien ASCII-Dateien, die in Textprogrammen bearbeitet werden können und umgekehrt.

Handlektüre

Das 226 Seiten starke Handbuch bietet zu allen FORMULARplus-Funktionen klare und ausführliche Informationen. Ein Index-Verzeichnis erleichtert die Suche nach einer bestimmten Problemlösung im Buch. Die Druckersteuerzeichen sind eingehend beschrieben. Zusätzlich wird eine hilfreiche Breitentabelle geliefert, die einem bestimmten Schriftattribut Zeichenzahl und Druckbreite zuordnet. Alle auf der Zusatzdiskette mitgelieferten Beispieldateien werden erläutert und durch einen Probeausdruck ergänzt. Die Mausfunktionen, Tastaturbelegung und Attribute sind in einer Hardcover-Karte zum schnellen Finden zusammengefaßt.

Fazit

FORMULARplus läuft auf ST- und TT-Computern in allen TOS-Versionen und auf Großbildschirmen. Für den Preis von 169,- DM erhält man ein ausgeklügeltes Programm, das voll zufriedenstellt und kaum Wünsche offen läßt. Die Funktionsvielfalt ist so gewaltig, daß es einige Zeit dauert, bis man alle Möglichkeiten überblickt. Wer aber ständig mit Daten und Formularen beschäftigt ist, nimmt die zahlreichen Arbeitserleichterungen gerne in Anspruch. FORMULARplus wird seinem hohen Anspruch gerecht und hat zudem ein günstiges Preis-Leistungsverhältnis.

Bezugsadresse:

Alfred Saß
Neuer Weg 2
W-2243 Albersdorf
Tel.: 04835/1447

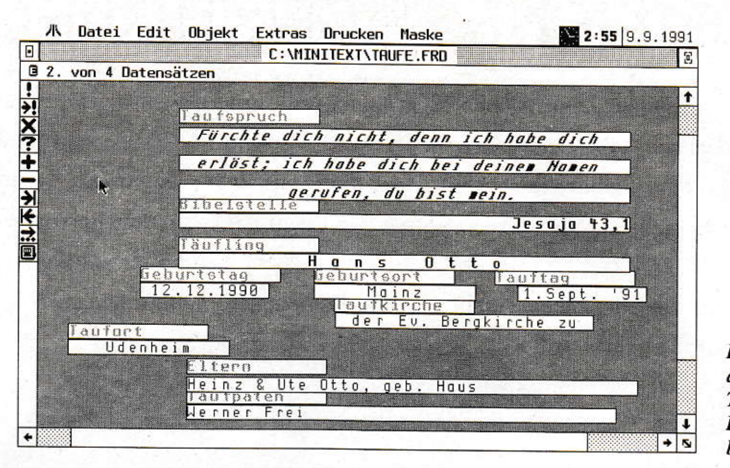
FORMULAR ST

Formulare erleichtern den bürokratischen Schriftverkehr. Täglich begegnen uns im öffentlichen Leben vorgedruckte Blätter, die nur an entsprechender Stelle ausgefüllt sein wollen. Warum sollte jemand, der einen Computer zu Hause stehen hat, nicht versuchen, sich die Bearbeitung solcher Formulare, angefangen bei einer Banküberweisung oder einem Scheck, so leicht und komfortabel wie möglich zu machen?! Dabei hilft ein Programm, das es versteht, bestimmte Felder zu definieren und auszufüllen.

Maskierungen

Eine Fileselectorbox erscheint nach dem Laden von FORMULAR ST (Version 2.20) zur Auswahl eines bereits erstellten Formulars. Wenn noch kein Formular vorhanden ist, drückt man den 'Abbruch'-Button. Im Menü 'Datei' unter 'Einstellungen' sollten nach dem ersten Programmstart wichtige Voreinstellungen, wie die Einstellung der Zugriffspfade für die diversen Dateien, der Farben für die Bildschirmdarstellungen und weiterer Parameter, vorgenommen werden. Die mausophilen Freunde des Atari werden mit diesem Programm garantiert auf ihre Kosten kommen. FORMULAR ST besitzt einen flexiblen Maskeneditor, mit dem beliebige Formulare mit ein Zehntel Positionsgenauigkeit erstellt werden können. Bei Anwählen des Punktes 'Neu anlegen' im Menü 'Maske' erscheint die Dialogbox 'Maße und Parameter', in der wichtige Voreinstellungen für die Erstellung eines Formulares getroffen werden: Unter 'Formulargröße' können ein festes Format von DIN-A6 bis DIN-A3 oder eine in Länge und Breite beliebige Größe vorgegeben werden. Für Endlospapier ist eine gesonderte Einstellung möglich. Des weiteren werden die Schriftgröße und der Maßstab der Bildschirmdarstellung der Maskenfelder sowie die Formelparameter und die Änderungsmöglichkeit der Maske als Objekt oder Maske eingestellt.

Nach dem Beenden der Dialogbox erscheint das Maskenfenster mit einer Funktionsleiste unterhalb der Menüleiste und einem horizontalen und vertikalen Millimetermaßband am Rand des Fensters. Vor der Eingabe der Textfelder ist zu bedenken, daß zu jedem Textobjekt ein Kommentarobjekt gehört. Das Kommentarobjekt wird automatisch mit dem Textobjekt in die Maske eingesetzt, wenn die Automatik in der Funktionsleiste eingeschaltet wurde. Das Kommentarobjekt benennt die jeweiligen Eingabefelder, um sie später bei der Dateneingabe wiederzuerkennen. Über einen Funktionsleisten-Button kann die Maske mit einem gleichmäßigen x-/y-Koordinatenraster belegt werden. Die erstellten Objekte werden dann nur auf den Rasterpositionen abgelegt, was bei Tabellenformularen sehr praktisch ist.



Die Strichlinien zeigen die Zuordnung der Text- zu den Kommentarobjekten bei FORMULAR ST

Nun endlich zur Maskenerstellung: Ein kurzer Klick mit der linken Maustaste auf den Knopf 'Textobjekt' hängt ein solches an den Mauszeiger und ist innerhalb des Maskenfenster an beliebiger Stelle mit erneutem Klick ablegbar. Während der Bewegung des Objektes zeigt das Objekt seine x- und y-Koordinaten an und wirft eine Verlängerungslinie zum Maßband am Fensterrand. Mit einem erneuten Klick auf ein bereits positioniertes Objekt in seinem linken Drittel ist dieses verschiebbar, um es an anderer Stelle neu abzulegen. Auf einen Mausklick im rechten Objektdrittel hin wird die Zeichenlänge angezeigt und veränderbar. So werden alle Objekte nacheinander erzeugt. Schiebt man die Objektlänge auf Null, erfragt eine Dialogbox, ob das Feld gelöscht werden soll. Als etwas ungünstig hat sich herausgestellt, daß die Funktionsleiste zu dicht an der Menüleiste liegt und letztere bei ungenauer Mausoperation aufgeklappt wird. Die Objekte mit der Maus zu holen, kann zum Geduldsspiel werden, wenn die Maus in die Jahre gekommen ist, und die Objekte nicht richtig am Mauszeiger hängen bleiben. Um jedoch nicht jedesmal mit der Maus in die Funktionsleiste fahren zu müssen, kann bei gedrückter Shift-Taste mit dem Mauszeiger aus dem bereits gesetzten Objekt ein neues mit gleicher Länge und Attributen gewonnen werden. Jedes gesetzte Objekt zeigt im Maskeneditor seine x-/y-Position und Länge an.

Der Arbeitsbildschirm zur Eingabe der Daten mit Icons zur Mausbedienung

Die Feinarbeit

Für die Feinpositionierung wird ein Objekt mit Doppelklick angewählt, und eine

Dialogbox bietet folgende Hilfen: Vertikale und horizontale Ausrichtung und Länge sind hier nachzubessern. Wenn nicht schon geschehen, gibt man für jedes Feld die gewünschte Schriftart und die Textpositionierung innerhalb des Objektes (rechts- und linksbündig oder zentriert) an. Ohne die Box zu verlassen, kann über Button zum nächsten Objekt gesprungen und ebensolche Einstellungen vorgenommen werden. Für die Gesamtzahl der Objekte läßt sich nachträglich eine vertikale und horizontale Verschiebung durch eine Randbestimmung erzielen.

Wenn beispielsweise drei Textobjekte mit einem Kommentarobjekt auskommen, weil sie dieselbe Bedeutung haben, darf zwischenzeitlich die automatische Kommentarobjektzuteilung in der Funktionsliste abgeschaltet werden. Dafür aber müssen nachträglich Zuordnungszeiger zwischen den Text- und dem Kommentarobjekt gesetzt werden, damit die vorübergehend kommentarlos bleibenden Textobjekte auch benannt sind. Die Zeiger lassen sich zur Überprüfung der Zuordnung anzeigen. Am Ende der Maskenerstellung kann über 'Objekt-Auswahl' die Reihenfolge der Felder mit Ziffern festgelegt werden, in der später die Dateneingabe erfolgt. Felder, die dabei ausgenommen werden sollen, erhalten keine oder eine höhere Nummer als vorhandene Formularfelder. Der Textinhalt eines Feldes kann im Menü 'Objekt' mit Hilfe einer Rubber-Linie kopiert, verschoben oder angehängt werden. Beim Kopieren und Verschieben geht der Text des Zielobjekts verloren, im letzten Fall auch der Inhalt im Quellobjekt. Beim Anhängen wird der Text des Quell- an das Zielobjekt angefügt.

Nach der Maskenerstellung wird der Maskeneditor verlassen und die Kommentartexte sind zunächst auf zweierlei Weise einzugeben: Entweder direkt im Eingabefenster oder komfortabler über einen Dialog-Handler, der Text- und Kommentarzeile anzeigt. Dabei muß unter 'Maße und Parameter' die Voreinstellung 'Objekt' anstatt 'Maske' getroffen worden sein. Bei Formularen, die über das Fenster hinausgehen, bietet sich die letztere Methode an. Im Dialog-Handler läßt sich zwischen den Objekten und Datensätzen hin- und herblättern.

Die Dateneingabe

Jetzt kann der erste Datensatz eingetippt werden. Der eingegebene Text wird mit Text- und Schriftattributen auf dem Bildschirm angezeigt, wie sie für das jeweilige Objekt vorgegeben wurden. Die Maske ist ohne Daten abspeicherbar. Die Datensätze aber lassen sich nur mit der Maske zusammen speichern. Dabei ist es möglich, ein Paßwort einzugeben, das beim erneuten Laden abgefragt wird. Zusätzlich erfolgt dann eine verschlüsselte Abspeicherung.

Am linken Rand des Dateneingabefensters finden sich die wichtigsten Funktionen für die Datensatzbehandlung auf mit der Maus direkt anklickbare Icons gelegt, die ein ständiges Herunterklappen der Menüleiste überflüssig machen. Eine Eingabeerleichterung bietet die Möglichkeit der Belegung von Funktionstasten, die separat abgespeichert wird. Die gerade geladene Tastenbelegung wird mit dem Formular abgespeichert und wieder geladen. Jede Funktionstaste darf vierfach mit bis zu 65 Zeichen belegt werden. Mit einer einfachen Suchen-Ersetzen-Funktion lassen sich beliebige Zeichenketten in den Datensätzen suchen und verändern.

Die Datensätze können anhand eines Sortierschlüssel, der bis zu sechs Textobjekte berücksichtigt, wahlweise alphanumerisch oder numerisch, in beliebiger Reihenfolge sortiert werden. Für jedes Textobjekt besteht die Möglichkeit, eine Rechenvorschrift zu definieren, deren Ergebnis als Objekttext übernommen werden kann. So sind die Grundrechenarten, Prozent, Maximum, Minimum, Durchschnitt, Rundung, Datum in drei Eingabeformen, automatische Summation, Stringoperationen, indirekte Indizierung und Rechenblatteingaben für allgemeine Formeln ausführbar. Bei Zeugnisformularen kann z.B. die Note aus den erreichten Punkten (also sieben bis neun Punkte ergibt die Note „befriedigend") berechnet werden. Die Formeleingabe in die Textfelder ist fast vollständig über Dialogbox mit der Maus erreichbar. Im Handbuch finden sich dazu ausführliche Erläuterungen.

FORMULAR ST ist in der Lage, ASCII-Texte zu im- und exportieren. Beim Datenimport wird die Anzahl der einzulesenden Datensätze mit der Angabe, wieviel Zeilen pro Datensatz berücksichtigt werden sollen, vorgewählt. Vor dem Importieren kann eine Objektauswahl der Datenfelder des bereits angelegten Formulars getroffen werden, in die eingelesen wird. Die zu importierenden Daten lassen sich so mit einem aktuellen Datensatz mischen. Am Dateianfang oder zwischen den Datensätzen lassen sich Bytes oder Zeilen angeben, die auszulassen sind. Der Exportvorgang schreibt den Inhalt jedes Objektes in eine neue Zeile, wobei jeder Datensatz durch eine Leerzeile getrennt wird. Auch hier können Objekte vorher in der bestehenden Datei vor dem Exportvorgang ausgewählt oder Datensätze ausgegrenzt werden.

Der Ausdruck

In der Druckeranpassung stehen die Sequenzen für die einzelnen Schriftarten und Sonderzeichen, den linken und oberen Rand, der beim Drucken ausgenommen bleibt, die Zeichen pro Zeile und die Zeichenbreite, Zeilenabstand und horizontalen Tabulator, die Initierung des Druckers. Alle diese Einstellungen können anhand des eigenen Druckerhandbuches verändert und abgespeichert werden. Ausgedruckt wird der aktuelle Datensatz unter Berücksichtigung der Objektauswahl, wenn diese gewählt wurde. Es sind aber auch die Anzahl der Drucke, von...bis Datensatz, mit oder ohne Kommentarobjekte, Spaltensatz ja oder nein anwählbar.

Das Handbuch

Das ca. 75seitige, geheftete Handbuch zu FORMULAR ST ist anhand der Menüfunktionen gegliedert und erklärt die einzelnen Programmöglichkeiten hinreichend und verständlich. Den Mausaktionen ist ein eigenes Kapitel gewidmet. Über das Indexverzeichnis erhält man im Problemfall einen schnellen Zugang zu den Erläuterungen. Das Handbuch muß noch der aktuellen Programmversion angepaßt werden. Im Programm sind Hilfstexte, die über die 'Help'-Taste erreichbar sind, integriert.

Fazit

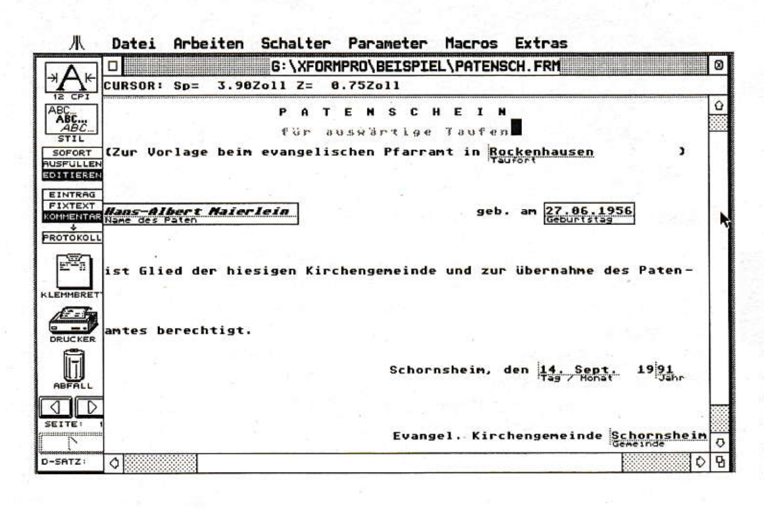
Das in GFA-BASIC 3.5 geschriebene FORMULAR ST läuft nur in der hohen ST-Auflösung und auf Großbildschirmen. Für den günstigen Preis von 89,-DM erhält der Anwender ein Programm, das ihm das Erstellen und Ausfüllen von Formularen praktisch erleichtert. Ich hätte mir gewünscht, daß manche Mausoperationen auch mit der Tastatur zu bedienen wären, um mit weniger Klicken auszukommen. Wer sich einmal in FORMULAR ST eingearbeitet hat, kommt sicherlich gut zurande und wird seine Freude haben.

Bezugsadresse:

ROSOFT
Stefan Rogel
Köhlerweg 1
6750 Kaiserslautern 31

XFORMPRO

Wer nach einer Möglichkeit gesucht hat, Felder in Formularen nacheinander mit dem Computer sofort zu bedrucken, muß nicht mehr seine alte Schreibmaschine belästigen und sich später darüber ärgern,



Der Arbeitsbildschirm zeigt die Arbeitsfläche mit Icons und das Arbeitsfenster.

daß er bei jedem neuen Formblatt mit der Arbeit wieder von vorne beginnen muß. Vorausgesetzt, ein Nadeldrucker ist vorhanden; denn mit einem Laserdrucker ist der Schreibmaschineneffekt nicht herzustellen.

Starten

Die Benutzeroberfläche von XFORMPRO teilt sich in das GEM-Textfenster als Arbeitsfläche für die Formularerstellung und ein Desktop, das Icons darstellt. Diese Icons machen die wichtigsten Funktionen bei der Formularerstellung mit der Maus zugänglich. Die Menüleistenfunktionen lassen sich sowohl mit der Maus als auch mit der Tastatur durchführen. Nach dem ersten Programmstart sind unter 'Parameter - Suchpfade' dieselbigen für alle wichtigen Dateien einzustellen.

Zurück zur Schreibmaschine...

Aus dem Computer wird nun eine Schreibmaschine: Man nehme ein Formblatt oder vorsichtshalber ein schnell mal selbstgezeichnetes und lasse es von einem Nadeldrucker einziehen, der daraufhin online sein muß. Im Anschluß an die Eröffnung einer (neuen) Datei ist auf dem Desktop das Symbol 'Sofortdruck' anzuklicken. In der Infozeile des Fensters ist dessen horizontale und vertikale Position - je nach Voreinstellung in Zoll, Zentimeter oder Millimeter - kontrollierbar. Wird der Cursor mit den Pfeiltasten in vertikaler Richtung bewegt, zieht der Drucker das Papier nach oben oder unten. In horizontaler Richtung bleibt der Druckkopf unbeeindruckt.

Bei Zuhilfenahme der Shift- oder Control-Taste wandert der Cursor in halben oder kleinen Schritten, aber nur, solange er ausgehöhlt erscheint. Unter 'Parameter-Zeilenabstände' sind diese Bewegungsschritte unterschiedlich einstellbar. Die horizontale Druckposition muß am Lineal des Druckes mit der Cursor-Postionsanzeige auf dem Arbeitsbildschirm mit Augenmaß abgestimmt werden.

Der mit Return abgeschlossene Eintrag wird nun sofort ausgedruckt. Doch es bedarf einiger Druckproben und ein wenig Geschicks, bis es gelingt, das Eingetippte an die richtige Formularstelle zu übertragen. Die Einzugtiefe - also der Abstand der ersten Druckzeile vom Papieranfang - sowie der Zeilenversatz - die Differenz des Druckkopfes zur Ausdruckposition - und der linke Offset - der Abstand des Druckrandes zum Blattanfang - müssen in einer Dialogbox bestimmt werden. Dabei ist darauf zu achten, daß der Zeilenausdruck über dem Schmetterling und unter dem Lineal des Druckers gelingt, um die Übersicht zu behalten. Mit einem Laserdrucker ist der Sofortdruck-Modus nicht funktionsfähig.

Solange der Cursor ausgehöhlt dargestellt wird, können Schriftgröße und -attribute über Icons für den nachfolgenden Eintrag jeweils neu bestimmt werden. Buttons zeigen verschiedene Textdefinitonen an: Ein 'Eintrag' - ein Text, der bei jedem Ausfüllen verändert wird - erscheint auf dem Bildschirm unterstrichen, ein 'Fixtext' - ein Feld mit immer gleichem Inhalt - normal, ein 'Kommentar' - ein nichtdruckbarer Hilfstext - hell und ein 'Protokoll' - Felder, die beim Ausdruck in eine extra Datei geschrieben werden - umrahmt. Einem gerade editierten Feld kann ein Label zugewiesen werden, das dem Feld seine unverkennbare Bezeichnung gibt. In der gleichen Box wird nach der maximalen Feldeingabelänge und Formartierung des Eintrags gefragt.

Wenn 'Attribute dynamisch' eingestellt ist, werden diese eintragsspezifisch beibehalten und sind nicht für alle Einträge des Formulars gleichzeitig bindend. Die vorgegebenen Formularränder erscheinen im Arbeitsfenster als gestrichelte Linien. Jedes Formular darf mehrere Seiten umfassen.

Den Sofortausdruck kann man auch unterdrücken. Man entwirft das Formular am Drucker, um nach Eingabe aller Formulartexte erst einen Probeausdruck auf ein leeres Blatt Papier vorzunehmen. Das „mühselig“ (Zitat aus dem Handbuch) erstellte Formular ist nach dem Ausdruck noch abzuspeichern, um jederzeit wiederverwendet zu werden.

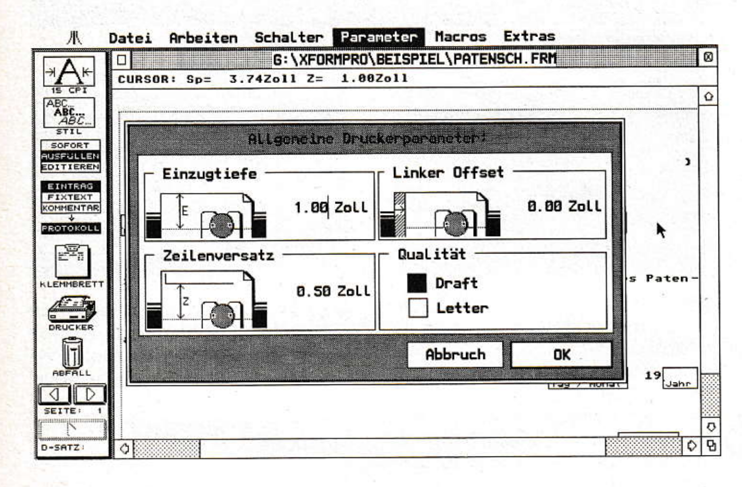
Editieren und ausfüllen

Im Editier-Modus wird ein Formular ohne Drucker erstellt. Die Feldpositionierungen sind entweder mit den Pfeiltasten oder über eine Dialogbox in der vorgewählten Maßeinheit einzugeben. Damit die Position des Cursors mit der Druckposition auf dem Papier übereinstimmt, muß die Einzugstiefe des Druckers exakt eingestellt sein. Selbstverständlich kann ein bereits erstelltes Formular in bezug auf seine Inhalte und Positionen der Felder verändert werden.

Wer ein Formular erstellt, hat sicherlich das Bedürfnis, es mit immer neuen Inhalten auszudrucken. Dazu reicht es, wenn ausschließlich die Feldinhalte gefüllt werden können, wie das im Ausfüll-Modus der Fall ist. Ist ein Formular geladen und sind seine Inhalte mit der Funktionstaste 'F4' gelöscht, kann man mit der Eingabe beginnen. Der Cursor springt nach Eingabe-Return oder Pfeiltastenbetätigung oder Anklicken eines Eintrages mit der Maus in das nächste Feld. Fixtexte und Kommentarfelder werden dabei übersprungen. Diese Einträge werden bei gedrückter Shift-Taste mit den Pfeiltasten angesprungen.

Ein Moduswechsel während der Arbeit mit XFORMPRO ist in der Regel möglich. In den Sofortdruck-Modus gelangt man aber nur, wenn ein Nadeldrucker 'online' geschaltet ist. Mit einem Laserdrucker funktioniert dieser Modus nicht. Wenn ein Eintrag entfernt wird, wechselt das Programm selbständig den Modus.

Ein Eintrag läßt sich mit der Maus auf den Mülleimer des Desktops ziehen und entfernen oder auf das Klemmbrett bewegen und damit kopieren oder zwischenspeichern. Über den Umweg des Klemmbrettes kann das Feld auf eine neue Position gezogen werden, indem man den Cursor entsprechend positioniert und das Feld aus dem Klemmbrett ins Textfenster zurückholt. Um Felder einzeln auszudrukken, können sie vom Fenster oder Klemmbrett her auf das Druckersymbol gebracht werden. Ein vergessenes Feld kann so nachträglich zu Papier gelangen.



Die allgemeinen Druckerparameter

Auswärtige Daten

XFORMPRO bietet eine einfache, aber flexibel funktionierende, universale Schnittstelle für Datenbanken. So können mit XFORMPRO erstellte Formulare mit Daten von außerhalb gefüllt werden. Wie geht das? Es ist eine Maske mit den gleichen Feldbenennungen und der gleichen Feldanzahl wie in der Datenbank zu erstellen. Fixtext- und Kommentar-Felder bleiben beim späteren Datentransfer unberücksichtigt. Im Menü 'Parameter' sind unter 'Parameter setzen' der Modus 'Synchronisation - automatisch', ein Einschlußzeichen für die Benennungsfelder und ein Trenn-String, der zwischen den Datensätzen gesetzt wird, zu definieren. Danach kann im Menü 'Datei - Db-Schnittstelle - Vorlage erstellen' nach Auswahl eines Dateinamens eine Textdatei mit den Bezeichnern der Felder erzeugt werden. Die Feldbezeichnungen werden darin- eingeschlossen vom Einschlußzeichen - untereinander geschrieben (#Name# etc.). Die Datenbank behandelt diese Datei wie einen Serienbrief und setzt anstelle der Bezeichner die Daten ein. Nach dem Laden des Formulars können die so gewonnenen Daten darin eingelesen und ausgedruckt werden.

Daten speichern und lesen

Ein einmalig erstelltes Formular kann mit hunderten von Datensätzen gelesen und gedruckt werden, wobei die Daten in einer eigenen Datei abgelegt und jeweils in die Vorlage eingelesen werden. Die eingetippten Daten speichert man in 'Datei: Daten... speichern'. Nach dem erneuten Ausfüllen des gleichen Formulars wird dieser Datensatz im Menü 'Datei: Daten...anhängen' in der gleichen Datendatei an den bereits gespeicherten Datensatz angehängt, und so weiter. Mit 'Datei: Daten...lesen' können die Daten eingelesen und wahlweise ausgedruckt werden, oder mit 'Datei: Daten-lesen&drucken' werden die Formulare der Reihe nach zum Drucker geschickt. Zum Hin- und Herblättern zwischen Datensätzen steht ein Icon im Desktop zur Verfügung.

Eine feine Sache ist die Protokoll-Funktion, die folgenden Sinn hat: Felder, die in einem Formular als Protokoll definiert und im Textfenster umrahmt dargestellt werden, speichert XFORMPRO beim Ausdruck des Formulars in einer Protokolldatei in der Form 'Label: Text' (also: Empfänger: ST-Computer, Betrag: 100,- DM usw.) extra ab. Diese Datei hilft der eigenen Vergeßlichkeit auf die Sprünge und bietet einen Überblick, beispielsweise über getätigte Überweisungen der zurückliegenden Wochen. In diese Protokollauszüge kann nach Voreinstellung das aktuelle Datum jedes Formularausdrucks geschrieben werden. An eine einmal angelegte Protokolldatei werden die neuen Protokollauszüge angehängt. Vorübergehend kann diese Funktion auch ausgeschaltet werden. Zur besseren Übersicht empfiehlt es sich, nur wenige Felder in einer Vorlage als Protokollfelder festzulegen.

Tastaturmakros

XFORMPRO bietet die Möglichkeit, jede Taste - mit und ohne Shift - zweifach zu belegen. Eine angelegte Makrodatei wird mit einem Formular automatisch nachgeladen. Dabei geht das Programm folgendermaßen vor:

Als erstes versucht XFORMPRO, unter dem eingestellten Pfad die Datei 'XFORMPRO.MAC' zu laden. Wird diese Datei nicht gefunden, sucht das Programm irgendeine MAC-Datei. Mißlingt auch dies, bleibt noch die Suche nach 'XFORMPRO.MAC' in dem Ordner, in dem das Programm steht. Es ist also sinnvoll, verschiedene Formular-Ordner anzulegen, in die man die entsprechenden Makrodateien ablegt.

Im Menü 'Makros - editieren' wird nach dem zu editierenden Buchstaben gefragt. Nach Tastendruck erscheint eine Dialogbox, in die ein Makro mit maximal vierzig Buchstaben eingegeben wird. Zusätzlich kann ein Vorschub definiert werden, der später dazu führt, daß nach dem Makroaufruf der Cursor in den nächsten Eintrag springt. Eine Übersicht über die bereits editierten Makros erhält man unter dem Menüpunkt 'Macros - Übersicht'. Bei Anklicken eines Buchstabens erscheint die Editierbox.

Aufgerufen werden alle Makros mit ESC plus Taste. Das automatische Einfügen von Datum und Zeit sowie die Belegung der Tasten Alternate-1 bis -0 mit Zahlwörtern ('eins', 'zwei'...) stehen als programmdefinierte Makros zur Verfügung.

Handliches

Das Handbuch erläutert die Schritte der Formularbehandlung auf dreiunddreißig Seiten hinreichend. An diese Erläuterungen schließt sich eine Bedienungsanleitung an, die in fünf Punkte untergliedert ist: 1. Eine Erklärung aller Cursor-Bewegungen, 2. eine Übersicht über die Tastaturbedienung, 3. die Bedeutung der einzelnen Menüpunkte, 4. eine Zusammenstellung der Desktop-Operationen und 5. ein Schaubild der in XFORMPRO benutzten Dateien.

Fazit

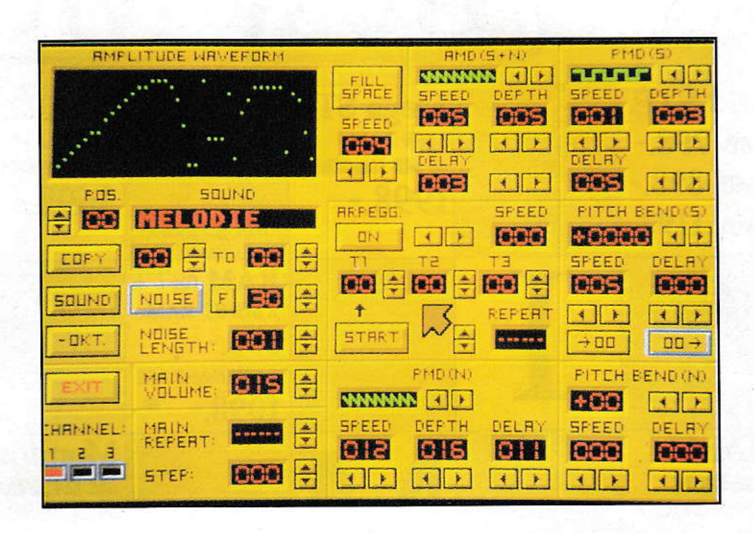
XFORMPRO läuft auf ST(E)-Rechnern mit mindestens 512 KB Speicher und Monochrommonitor wie unter der hohen TT-Auflösung, unter Autoswitch-Overscan und verschiedenen Grafikkarten. Das Programm unterstützt alle Nadel-, Typenrad- und Laserdrucker, für die eine WordPlus-Druckeranpassung zu bekommen ist. Der Einführungspreis von XFORMPRO beträgt 159,- DM und ist m.E. im Preis-Leistungsvergleich mit ähnlichen Programmen eine Idee zu hoch gegriffen. XFORMPRO besticht durch seine einfache Bedienung. Die Möglichkeit, ein Formular direkt am Nadeldrucker auszufüllen, gehört zu seinen Stärken und Besonderheiten, obgleich es ein wenig Übung bedarf, diese Arbeitsweise auch wirklich zu beherrschen.

Kurt Rainer Klein

Bezugsadresse:

Richter-Distributor
Hagener Straße 65
5820 Gevelsberg
Tel.: 02332/2706

Let the music play



Musik Mon-ST, macht dem Soundchip Beine

Wer kennt das nicht ? Da hat Man(n) oder Frau ein tolles Programm geschrieben, das nun auf die ST-Menschheit losgelassen werden soll. Alles stimmt, die Grafik, der Programmfluß, die GEM-Fenster, der Sound ...!
Halt, der Sound ??
Statt einem fetzigen Rhythmus werden dem Soundchip nur ein paar mickrige Töne entlockt, die das Programm ohne Zweifel um ein paar Prozentpunkte in der Gunst der Anwender sinken lassen. Nun ist guter Rat teuer. Die meisten Hobbyprogrammierer werden vor schier unlösbare Probleme gestellt, dieses Manko zu beheben, denn hier hilft nur Assembler-Programmierung weiter.

Doch welcher Hobbyprogrammierer hat schon die Zeit und die Geduld, sich in die Tiefen des ST hinabzubegeben und den ultimativen Kampf der Bits und Bytes aufzunehmen? Bis vor kurzem konnte ich mich ebenfalls zu diesem Kreis zählen, bis mir eine gütige Fügung des Schicksals das Programm MUSIC MON ST zutrug und den Dornröschenschlaf des Soundchips beendete.

Fantastische Klänge

MUSIC MON ist ein Soundeditor, der fantastische Klänge erzeugt, die über Editor oder MIDI (Musical Instruments Digital Interface) in einfachster Weise eingegeben, per Sequenzer zu kompletten Musikstücken zusammengesetzt und über eine Assembler-Routine in eigene Programme eingebunden werden können. Das Programm lag mir in der Version 1.1 vor und ist nicht kopiergeschützt, so daß problemlos Arbeitskopien erstellt werden können. Auch einer Installation auf Festplatte steht nichts im Wege. Die ca. 60 Seiten starke Programmbeschreibung erläutert in selbst für Musiklaien einfachster Weise die komplette Bedienung und wird in einer Programmhülle mit Ringbuchlochung mitgeliefert. Vor dem Start von MUSIC MON hat der Anwender die Wahl zwischen der Farb- oder der Monochromversion. In der Farbversion ist das Programm hübsch anzuschauen und wesentlich aussagekräftiger, alle weiteren Ausführungen werden sich auf die Monochromversion beziehen. Nach dem Laden erscheint ein Titelbild mit einem flotten Musikstück, das den Anwender schon vorab die Leistungsfähigkeit dieses Programms erahnen läßt. Per Mausklick öffnet sich der Hauptbildschirm, auf dem sich alle Funktionen zum Erstellen und Verändern eines Musikstücks befinden. Anfangs verwirrt die Vielfalt der Buttons und Anzeigen, doch erkennt man auf den zweiten Blick die wohlgeordnete Logik. Spätestens jetzt fällt dem Anwender auch die fehlende Menüleiste auf. Accessories können daher nicht angesprochen werden. Dieses Manko ist nicht mehr ganz zeitgemäß und sollte in einer späteren Version geändert werden.

Noteneingabe

Im unteren Teil des Bildschirms befindet sich das Notenanzeigefeld, über das ein Musikstück eingegeben und editiert werden kann. Das Notenanzeigefeld besteht aus einer Positionsanzeige und drei Spalten, für jeden Soundkanal eine. Die Noten können über den MIDI-Kanal oder über die Tastatur eingegeben werden, wobei die ST-Tastatur hier nun die Funktion einer Klaviatur erhält. Mit den Funktionstasten läßt sich die gewünschte Oktave auswählen, die in einem kleinen Rechteck neben dem Notenanzeigefeld angezeigt wird. Ein Eintrag in die Notenliste kann auf zwei Wegen erfolgen. Entweder spielt man auf der ST-Tastatur die entsprechende Note oder man gibt die Note manuell in der Reihenfolge Note, Oktave und Soundnummer ein. Die Notenlänge ergibt sich aus dem Abstand von einer Note zur nächsten, wobei jede Zeile in der Notenliste eine 16tel-Notenlänge darstellt. Die Musiker unter ihnen werden jetzt sicherlich anmerken, daß eine 16tel Notenauflösung

nicht gerade viel ist. Stimmt, man hat aber die Möglichkeit, die Abspielgeschwindigkeit in weitem Rahmen zu variieren. Durch Verdoppeln der Abspielgeschwindigkeit und entsprechendes Umformatieren der Notenwerte lassen sich z.B. schon Notenauflösungen bis 32tel erzielen. Die Soundnummer gibt an, mit welchem Instrument die Note gespielt wird. Für jedes Musikstück können maximal 50 verschiedene Sounds (Instrumente) verwendet werden.

Der Sequenzer

Die wichtigste Funktion von MUSIC MON ST ist der Sequenzer, mit dem in Baukastenmanier ein Musikstück zusammengebastelt werden kann. Ein komplettes Musikstück setzt sich aus einzelnen Teilstükken, sogenannten Patterns, zusammen, von denen 70 Stück zur Verfügung stehen. Der Vorteil einzelner Patterns liegt darin, daß sich diese beliebig oft in einem Musikstück wiederholen lassen, ohne daß dieses Teilstück erneut komponiert werden muß. Ein Pattern hat eine maximale Länge von 4 Takten, die aber individuell verändert werden kann. Möchte man nun die einzelnen Patterns zu einem kompletten Song zusammenfügen, so trägt man die Nummern der entsprechenden Patterns in die Sequenzerliste ein, die Platz für maximal 100 Einträge bietet. MUSIC MON bietet nun drei verschiedene Abspielfunktionen. Zum einen kann das momentan eingestellte Pattern abgespielt werden, zum anderen der komplette Song gemäß der Sequenzerliste. Ferner lassen sich alle Patterns zwischen dem eingestellten Start-Pattern und End-Pattern abspielen. Selbstverständlich verfügt MUSIC MON auch über leistungsfähige Edit-Funktionen, mit denen sich einzelne Patterns oder Sounds kopieren oder auch Noten transponieren lassen. Hat der Anwender nun ein Musikstück erstellt, möchte er es verständlicherweise auf Diskette sichern. Hier bietet MUSIC MON die Wahl zwischen dem Speichern einzelner oder aller Patterns und Sounds sowie dem erstellten Song gemäß der Sequenzerliste. Ferner kann hier das komponierte Musikstück in einem eigenen kompakten Format gespeichert werden, das man dann in eigene Programme zusammen mit der Abspielroutine einbinden kann. Diese Funktion weist seltsamerweise in der Monochromversion einen Fehler auf, der aber in der nächsten Version, so wurde mir von Galactic versichert, beseitigt sein wird.

Der Soundbildschirm

Der zweite wichtige Bildschirm von MUSIC MON ST ist der Soundbildschirm, dessen Anblick dem Anwender beim ersten Mal den Atem verschlägt. Hier werden alle Möglichkeiten zur Manipulation von Sounds angeboten, die das Herz des Musikfreundes höher schlagen läßt. Durch Betätigung eines angeschlossenen MIDI-Keyboards oder durch Druck auf die Atari-Tastatur kann der bearbeitete Sound natürlich sofort mitgehört werden. Links oben auf dem Bildschirm stellt der Anwender in einem Rasterfeld mit der Maus die Hüllkurve ein, die den grundlegenden Lautstärkeverlauf der Note festlegt. Weiter unten kann man angeben, ob sich die Rauschfrequenz in Abhängigkeit von den gespielten Noten ändern soll oder nicht. Eine sehr wichtige Funktion ist die Definition der Frequenzmodulation, unter der man die ständige leichte Veränderung der aktuellen Tonfrequenz versteht. Hier stehen Dreieck-, Rechteck- und zwei Sägezahnformen zur Auswahl. Möchte man die aktuelle Frequenz um einen gewissen Grad nach oben oder unten gleiten lassen, so hilft die Funktion Pitch Bend, zu deutsch Tonhöhenverbiegung, weiter. Selbstverständlich gelten die beiden letzten Funktionen auch für die Rauschgeneratoren. Unter der Amplitudenmodulation versteht die Musikwelt die regelmäßige Veränderung der aktuellen Lautstärke, deren Bedienungselemente sich hierfür ganz oben auf dem Bildschirm neben der Frequenzmodulation für den Tongenerator befinden. Last but not least bietet MUSIC MON auch ein Arpeggio, d.h. eine schnelle, sich ständig wiederholende Abfolge von einzelnen Tönen, womit sich beispielsweise Akkorde simulieren lassen. Verständlicherweise kann hier nur in Kurzform auf diese umfangreichen Funktionen eingegangen werden.

Soundeinbindung

Die Musikstücke, die mit MUSIC MON erstellt wurden, lassen sich in jede Programmiersprache einbinden, die eine Funktion zum Aufruf von Assembler-Unterprogrammen besitzt. Auf der Programmdiskette werden Beispiele für GFA-BASIC, Omikron-Basic und Assembler mitgeliefert. In GFA-BASIC erfolgt die Einbindung absolut unproblematisch: Die Soundroutine und das Musikstück werden über die BASIC-Funktion 'BLOAD' an eine Adresse eingeladen und mittels 'C:' aufgerufen. Die Soundroutine hängt sich in den VBL-Interrupt ein und klinkt sich nach Beendigung des Musikstücks wieder selbständig aus. Ich möchte noch erwähnen, daß MUSIC MON vier verschiedene Soundroutinen anbietet, die sich unterschiedlich in den Interrupt einhängen. Dies ist besonders nützlich, falls andere Programme oder Accessories sich ebenfalls der Interrupt-Programmierung bedienen und dadurch mit der Abspielroutine kollidieren könnten.

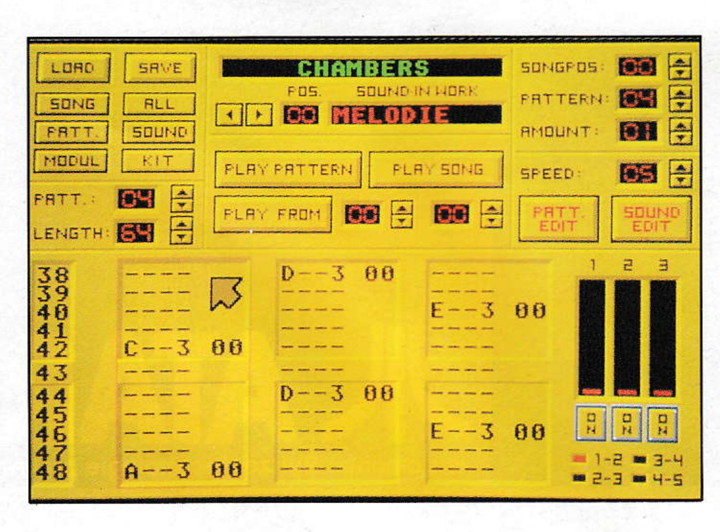
Fazit

MUSIC MON ST ist ein Programm, das vielen Anwendern wieder Hoffnung geben wird, die eigene Musikstücke in ihre Programme einbinden möchten. Auch wenn die Bedienung anfangs etwas ungewohnt ist, weil für die Noteneingabe kein richtiges Notenblatt zur Verfügung steht, sondern die Noten quasi textähnlich eingegeben werden, findet sich der Anwender dennoch nach kurzer Zeit gut zurecht. Wenn in der nächsten Version der Bug beim Speichern in der monochromen Auflösung entschärft worden ist, steht den ST-Usern ein rundum gelungenes Programm zur Verfügung.

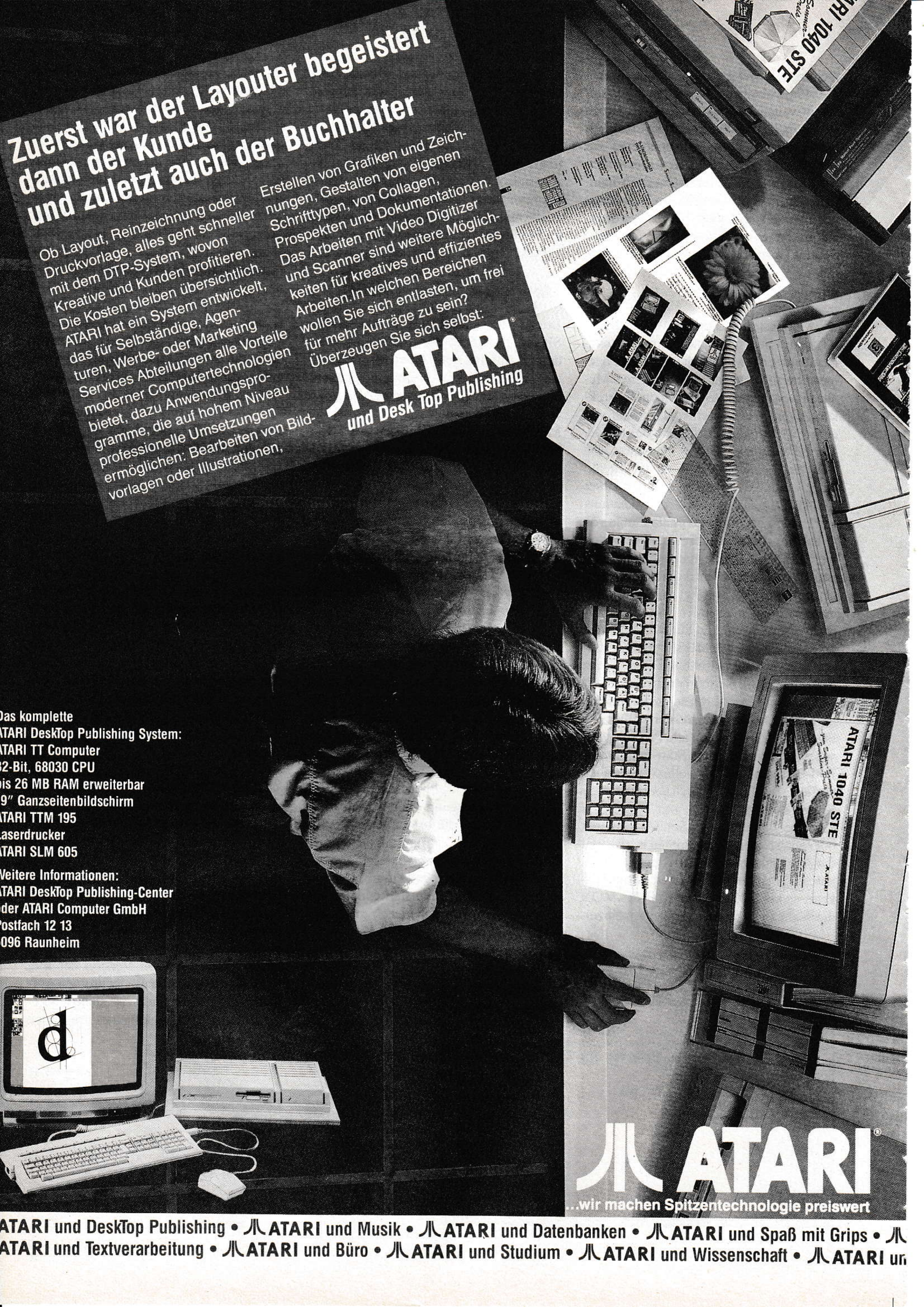
Rainer Wolff

Bezugsquelle:

Galactic
Julienstr. 7
W-4300 Essen 1
Tel.:(020) 792081



Der Sequenzer erleichtert das Einspielen von ganzen Songs erheblich.



Zuerst war der Layouter begeistert
dann der Kunde
und zuletzt auch der Buchhalter
Ob Layout, Reinzeichnung oder Druckvorlage, alles geht schneller mit dem DTP-System, wovon Kreative und Kunden profitieren. Die Kosten bleiben übersichtlich. ATARI hat ein System entwickelt, das für Selbständige, Agenturen, Werbe- oder Marketing Services Abteilungen alle Vorteile moderner Computertechnologien bietet, dazu Anwendungsprogramme, die auf hohem Niveau professionelle Umsetzungen ermöglichen: Bearbeiten von Bildvorlagen oder Illustrationen, Erstellen von Grafiken und Zeichnungen, Gestalten von eigenen Schrifttypen, von Collagen, Prospekten und Dokumentationen. Das Arbeiten mit Video Digitizer und Scanner sind weitere Möglichkeiten für kreatives und effizientes Arbeiten. In welchen Bereichen wollen Sie sich entlasten, um frei für mehr Aufträge zu sein? Überzeugen Sie sich selbst:
ATARI®
und Desk Top Publishing
Das komplette
ATARI DeskTop Publishing System:
ATARI TT Computer
32-Bit, 68030 CPU
bis 26 MB RAM erweiterbar
19" Ganzseitenbildschirm
ATARI TTM 195
Laserdrucker
ATARI SLM 605
Weitere Informationen:
ATARI DeskTop Publishing-Center
oder ATARI Computer GmbH
Postfach 12 13
6096 Raunheim
ATARI 1040 STE
ATARI®
...wir machen Spitzentechnologie preiswert
ATARI und DeskTop Publishing • ATARI und Musik • ATARI und Datenbanken • ATARI und Spaß mit Grips •
ATARI und Textverarbeitung • ATARI und Büro • ATARI und Studium • ATARI und Wissenschaft • ATARI un

Programmier Praxis

INHALT

OBJEKTIV

Uwe Seimet

Nachdem im letzten Heft Routinen vorgestellt wurden, um die Extended Rectangle Library des GEM/3 für IBM-kompatible PCs in Pure C auf dem Atari zu realisieren, soll es nun darum gehen, auch die Extended Object Library umzusetzen.

Diese Bibliothek ermöglicht eine vereinfachte Verwaltung von Dialogen und stellt darüber hinaus Funktionen zur Verfügung, um einzelne Objekt-Flags sowie den Objektstatus gezielt manipulieren zu können. Um welche Routinen es sich handelt? Nun, hier eine Auflistung:

void ob_dostate(OBJECT*tree, int index, int state);
Mit Hilfe von *ob_dostate* ist es möglich, gezielt das Status-Flag state im Objekt *index* des Objektbaums *tree* zu setzen.

void ob_undostate(OBJECT *tree, int index, int state);
ob_undostate ist das Gegenstück zu *ob_dostate*. Es wird also ein bestimmtes Flag zurückgesetzt.

int ob_isstate(OBJECT *tree, int index, int state);
Dieser Aufruf ermöglicht die Abfrage eines Status-Bits. Das Ergebnis ist TRUE, wenn das angesprochene Bit state gesetzt ist, andernfalls erhält man FALSE.

void ob_doflag(OBJECT *tree, index, int flag);
ob_doflag erlaubt das Setzen eines einzelnen Objekt-Flags.

void ob_undoflag(OBJECT *tree, index, int flag);
Um ein Flag selektiv zurückzusetzen, steht *ob_undoflag* zur Verfügung.

int ob_isflag(OBJECT *tree, index, int flag);
Mit *ob_isflag* kann getestet werden, ob ein Objekt-Flag gesetzt ist. In diesem Fall wird TRUE zurückgeliefert.

int ob_xywh(OBJECT *tree, int index, GRECT *prect);
ob_xywh liefert die Ausmaße eines Objekts in Form einer GRECT-Struktur, die bereits im letzten Heft in Verbindung mit der Extended Raster Library angesprochen wurde.

char *ob_get_text(OBJECT *tree, int index, int clear);
Diese Funktion liefert einen Pointer auf einen Text-String zurück, falls das angegebene Objekt einen solchen enthält. Es muß sich also um ein Objekt des Typs G_TEXT, G_FTEXT, G_BOXTEXT, G_FBOXTEXT, G_STRING,G_BUTTON oder G_TITLE handeln.

Ist das Flag clear TRUE, wird das erste Zeichen des Strings zusätzlich durch eine Null überschrieben, was sich ausgezeichnet dazu eignet, Eingabefelder vor einem Dialog zu löschen.

char *ob_set_text(OBJECT *tree, int index, char *ptr);
Ein neuer Textzeiger kann mit *ob_set_text* gesetzt werden. ptr zeigt in diesem Fall auf den String, der Bestandteil der Objektstruktur werden soll.

int ob_draw_dialog (OBJECT *tree, int x, int y, int w, int h);
ob_draw_dialog übernimmt das komplette Zeichnen einer zentrierten Dialogbox. Sind die Parameter x,y,w,h ungleich Null, wird zusätzlich eine sich bis auf Dialoggröße ausbreitende *growbox* mit den ensprechenden Startkoordinaten gezeichnet.

int ob_undraw_dialog(OBJECT *tree, int x, int y, int w, int h);
Die Funktion dieses Aufrufs liegt auf der Hand. Nach Beendigung eines Dialogs kann optional eine *shrinkbox* gezeichnet werden, wenn die Parameter x,y,w,h die Koordinaten hierzu enthalten.

Alle Funktionsprototypen sind noch einmal im Listing von OBLIB.H zusammengefaßt.

Den Quellcode für die Extended Object Library enthält OBLIB.S. Assembliert wurde dieser Text mit dem MAS-Assembler. Natürlich eignet sich hierzu auch ein anderer Assembler. Spezifisch für Pure C ist die Übergabe der Funktionsparameter.

Die Verwendung der neuen Library-Routinen wird im Listing DIALOG.C demonstriert. *ob_draw_dialog* ersetzt alle Aufrufe von *form_center* und *form_dial*. Das Rücksetzen des SELECTED-Status beim Exit-Button eines Dialogs erfolgt der Einfachheit halber mittels *ob_undostate*.

```
/*****************************************/
/* OBLIB.H Allgemeine OBLIB Definitionen */
/*****************************************/
/*                                       */
/*    Autor:  Uwe Seimet                 */
/*         (c) 1991 MAXON Computer       */
/*****************************************/

#ifndef __OBLIB__
#define __OBLIB__

void ob_dostate ( OBJECT *tree, int index, int state );
void ob_undostate ( OBJECT *tree, int
                  index, int state );
int ob_isstate ( OBJECT *tree, int index, int state );
void ob_doflag ( OBJECT *tree, int index, int flag );
void ob_undoflag ( OBJECT *tree, int index,int flag );
int ob_isflag ( OBJECT *tree, int index, int flag );
```
→

```
void ob_xywh ( OBJECT *tree, int index,
               GRECT *rec );
char *ob_get_text ( OBJECT *tree, int
                    index, int clear );
void ob_set_text ( OBJECT *tree, int index,
                   char *p );
void ob_draw_dialog ( OBJECT *tree, int xl,
                      int yl, int wl, int hl );
void ob_undraw_dialog ( OBJECT *tree,
                int xl, int yl, int wl, int hl );

#endif /* __OBLIB__ */
```

```
; OBLIB.S
; (c) 1991 MAXON Computer

          globl ob_dostate
          globl ob_undostate
          globl ob_isstate
          globl ob_doflag
          globl ob_undoflag
          globl ob_isflag
          globl ob_xywh
          globl ob_get_text
          globl ob_set_text
          globl ob_draw_dialog
          globl ob_undraw_dialog

          globl form_center
          globl form_dial
          globl objc_draw

G_TEXT    = 21
G_BOXTEXT= 22
G_BUTTON = 26
G_STRING = 28
G_FTEXT   = 29
G_FBOXTEXT= 30
G_TITLE   = 32


ob_dostate:
          mulu #24,d0
          or d1,10(a0,d0)     ;Statusbit
                               setzen
          rts


ob_undostate:
          not d1
          mulu #24,d0         ;Statusbit
                               zurücksetzen
          and d1,10(a0,d0)
          rts


ob_isstate:
          mulu #24,d0
          and 10(a0,d0),d1    ;Statusbit
                               testen
          bne sset
          clr d0
          rts
sset:     moveq #1,d0
          rts


ob_doflag:
          mulu #24,d0
          or d1,8(a0,d0)      ;Objekflag
                               setzen
          rts


ob_undoflag:
          not d1
          mulu #24,d0
          and d1,8(a0,d0)     ;Objektflag
                               setzen
          rts


ob_isflag:
          mulu #24,d0
          and 8(a0,d0),d1     ;Objektflag
                               testen
          bne fset
          clr d0
          rts
fset:     moveq #1,d0
          rts


ob_xywh:
          mulu #24,d0
          move.l 16(a0),(a1)+
          move.l 20(a0),(a1)
          rts


ob_get_text:
          bsr typetest        ;auf Objekttyp
                               testen
          move.l (a0),a0
          tst d1              ;String
                               initialisieren?
          beq noinit          ;nein-
          clr.b (a0)
noinit:   rts


ob_set_text:
          bsr typetest
          move.l a1,(a0)      ;neuen
                         Stringpointer setzen
          rts


typetest: mulu #24,d0
          move.b 7(a0,d0),d2  ; Objekttyp
          cmp.b #G_TEXT,d2
          beq text
          cmp.b #G_FTEXT,d2
          beq text
          cmp.b #G_FBOXTEXT,d2
          beq text
          cmp.b #G_BOXTEXT,d2
          beq string
          cmp.b #G_STRING,d2
          beq string
          cmp.b #G_BUTTON,d2
          beq string
          cmp.b #G_TITLE,d2
          beq string
          sub.l a0,a0
          addq.l #4,sp        ; Abbruch, falls
                               kein Textobjekt
          rts
text:     move.l 12(a0,d0),a0
          rts
string:   lea 12(a0,d0),a0
          rts


ob_draw_dialog:
          link a6,#-16
          bsr center
          clr -2(a6)          ;FMD_START
          bsr dial
          move -4(a6),d0      ;x
          or -6(a6),d0        ;y
          or -8(a6),d0        ;w
          or 8(a6),d0         ;h
          beq nogrow          ;keine growbox
                               zeichnen-
          move #1,-2(a6)      ;FMD_GROW
          bsr dial
nogrow:   lea -16(a6),a1      ;Pointer auf
                         zentrierte Koordinaten
          move (a1)+,-(sp)    ;h
          move (a1)+,-(sp)    ;w
          move (a1)+,-(sp)    ;y
          move (a1),d2        ;x
          moveq #127,d1       ;alle Ebenen
                         (127 sollten reichen)
          clr d0              ;ab Ebene 0
          bsr objc_draw
          addq.l #6,sp
          unlk a6
```

→

```
        rts


ob_undraw_dialog:
        link a6,#-16
        bsr center
        move -4(a6),d0      ;x
        or -6(a6),d0        ;y
        or -8(a6),d0        ;w
        or 8(a6),d0         ;h
        beq noshrink        ;keine
                      shrinkbox zeichnen-
        move #2,-2(a6)      ;FMD_SHRINK
        bsr dial
noshrink:move #3,-2(a6)     ;FMD_FINISH
        bsr dial
        unlk a6
        rts


center:
        move d2,-8(a6)      ;fo_dilittlw
        move d1,-6(a6)      ;fo_dilittly
        move d0,-4(a6)      ;fo_dilittlx
        move.l a0,-(sp)
        lea -10(a6),a1
        pea -16(a6)
        pea -14(a6)
        pea -12(a6)
        bsr form_center
        lea 12(sp),sp
        move.l (sp)+,a0
        rts

dial:
        move.l a0,-(sp)     ;Objektadresse
                             merken
        lea -16(a6),a1      ;Pointer auf
                      zentrierte Koordinaten
        move (a1)+,-(sp)    ;fo_dibigh
        move (a1)+,-(sp)    ;fo_dibigw
        move (a1)+,-(sp)    ;fo_dibigy
        move (a1),-(sp)     ;fo_dibigx
        move 8(a6),-(sp)    ;fo_dilittlh
        move -8(a6),-(sp)   ;fo_dilittlw
        move -6(a6),d2      ;fo_dilittly
        move -4(a6),d1      ;fo_dilittlx
        move -2(a6),d0      ;fo_diflag
        bsr form_dial
        lea 12(sp),sp
        move.l (sp)+,a0
        rts
```

```
/* Dialog mittels ob_draw_dialog,
   ob_undraw_dialog und ob_undostate         */
/* tree erhält die Adresse des Objektbaums */
/* index ist die Nummer des ersten
   Eingabefeldes                             */

void dialog(tree,index)
OBJECT *tree;
int index;
{
  int exit_obj;

  ob_draw_dialog(tree,40,40,4,4);
  /* Dialog mit growbox zeichnen    */
  exit_obj=form_do(tree,index);
  /* Benutzereingaben abwarten      */
  ob_undraw_dialog(tree,40,40,4,4);
  /* Dialog entfernen, shrinkbox   */
  ob_undostate(tree,exit_obj,SELECTED);
  /* Exit-Button zurücksetzen       */
  return;
  /* so einfach ist das             */
}
```

"FLIEGENDE DIALOGE" IN GFA-BASIC

Wolfgang Kohlmaier

DIALOGBOXEN HABEN DIE UNSCHÖNE ANGEWOHNHEIT, IMMER DORT ZU ERSCHEINEN, WO SIE DIE GERADE BENÖTIGTEN INFORMATIONEN ZUR EINGABE VERDECKEN. 'FLIEGENDE DIALOGE' KÖNNTEN DIESES PROBLEM ELEGANT UMGEHEN, ABER LEIDER EXISTIEREN BISLANG NUR LÖSUNGEN IN 'C'. DEN BASIC-PROGRAMMIERERN BLIEBEN SOMIT NUR DER BISS IN DEN SAUREN APFEL UND DER VERZICHT AUF SOLCHE ANNEHMLICHKEITEN. DIESES MANKO SOLL MIT DEN IM FOLGENDEN VORGESTELLTEN ROUTINEN BESEITIGT WERDEN.

Die Fähigkeiten der originalen GEM-Dialogboxen sind, was die Flexibilität ihrer Positionierung am Bildschirm angeht, recht verkümmert. Zumindest erscheint es im ersten Augenblick so. In Urzeiten des Atari ST war die zentrierte Ausgabe am Bildschirm meistens die einzige Darstellungsvariante, die in Programmen zum Ausdruck kam. Allzu oft wurden dabei Informationen, gerade in dem Augenblick, in dem sie benötigt wurden, von der Dialogbox verdeckt. Also 'ABBRUCH' im Dialog anwählen (sofern der Programmierer daran gedacht hat) und darunter nachsehen und merken (na, wozu hat man denn im Computerzeitalter einen Notizblock?).

Mit zunehmender Verbreitung der Großbildschirme wurde eine weitere Schwachstelle dieser zentrierten Dialoge evident. Auch wenn der Hauptarbeitsaufwand - beim Hantieren in der Menüleiste - in der linken oberen Ecke lag, mußte man aufgrund der zentrierten Dialogboxen regelmäßig einen Mausmarathon hinlegen, um bei einer Abfrage die gewünschten Knöpfe zu erreichen.

Das Problem blieb nicht lange unbeachtet, und schon bald

```
Flying Dial's Demonstration
Eingabe1:______________          Demo
Eingabe2:______________          Abbruch

Mover - Bewegungszeichen     Grundrahmen=Objekt Nul
                             wird als
                             TOUCHEXIT definiert
```

Dialogbox zur Demonstration

war die Idee zu beweglichen Boxen geschaffen. Diese als C-Bibliothek vorliegenden FLYDIALs wurden von Julian F. Reschke entworfen und programmiert. In der Januarausgabe des heurigen Jahres wurde ergänzend auch die 'MOVEDIAL'-Bibliothek in der ST-Computer vorgestellt. Mittlerweile gehört es - zumindest in professionellen Programmen - fast schon zum guten 'Programmierton', sich dieser FLYDIALs zu bedienen.

Das ursprüngliche Problem blieb letztendlich den Programmierern in GFA-BASIC erhalten, denn sie konnten die C-Routinen nicht nutzen. Aus diesem Grund möchte ich Ihnen eine Abwandlung der AES-Routine *FORM_DO* vorstellen, die unter anderem ein Verschieben der Dialogboxen zuläßt. Diese Funktion wurde für den Einsatz in GFA-BASIC-Programmen entwickelt. Eine Umsetzung in OMIKRON BASIC dürfte aber keine Probleme bereiten.

Welche Routinen werden zur Verfügung gestellt?

1)FORM_DO(Baum,Objekt, x-Koordinate,y-Koordinate)

2)FORM_DRAW als Subroutine von FORM_DO

Die gesamte Arbeit übernimmt eine GFA-Funktion mit dem Namen *FORM_DO()*, die in einer erweiterten Parameterliste auch die Dialogboxkoordinaten übernimmt. Die Prozedur *Form_draw* wird nur als Subroutine von *FORM_DO()* eingesetzt.

Das AES bietet Ihnen - um den Verwaltungsaufwand bei Formularen gering zu halten - die fix und fertige Funktion FORM_DO() an. Diese Routine übernimmt, nach Darstellung einer Dialogbox, die vollständige Abarbeitung derselben. Die Originalroutine erfordert die Angabe des Objektbaumes und die Nummer des ersten Objektes, ab dem die Bearbeitung erfolgen soll. Als Funktionsrückgabe erhält man die Nummer des Objektes, über welches das Formular verlassen wurde. Das Verlassen eines Objektes mittels Doppelklick wird durch Setzen des

höchsten Bits angezeigt. In diesem Fall muß man, um zur Objektnummer zu gelangen, das Bit ausmaskieren, was durch die GFA-Konstruktion *form&= (form& and &HFF)* erreicht wird.

Die modifizierte *FORM_DO*-Funktion unterscheidet sich in zwei Punkten vom Original. Erstens können die x- und y-Koordinaten an die Funktion übergeben werden, um das Formular an den entsprechenden Koordinaten darzustellen. Da diese Variablen als sogenannter 'Call by reference' an die Funktion übergeben werden, können Koordinatenveränderungen - die z.B. durch Verschieben des Formulares bewirkt werden - an die aufrufende Prozedur zurückgegeben werden. Sollten sowohl die x- als auch die y-Koordinate undefiniert sein (entspricht dem Wert Null), wird automatisch eine Zentrierung durchgeführt. Diese können Sie auch jederzeit aktiv durchführen, indem Sie das Bewegungssymbol (MOVER) mit einem Doppelklick anwählen. Danach wird Ihre Dialogbox automatisch zentriert erscheinen.

Wieso läßt sich nun eine Dialogbox plötzlich verschieben? Die Frage wird dann klar, wenn man sich die verschiedenen Status- und Flaggendefinitionen (oder besser Flags) der RSC-Objekte näher ansieht. Was benötigt man zur Realisierung einer Verschieberoutine? Zuerst muß man den Austieg aus der AES-Routine *FORM_DO* schaffen. Dies funktioniert mit Hilfe eines kleinen Tricks. Das Grundobjekt, welches den Boxrahmen darstellt, hat die Objektnummer Null. Man muß also das nullte Objekt so definieren, daß bei Mausklick darauf die *FORM_DO*-Routine verlassen wird. Zu diesem Zweck gibt es das Objekt-Flag *'TOUCHEXIT'*, welches man am Objekt Null aktivieren muß. In GFA-BASIC sieht das folgendermaßen aus (s. Bild):

```
tree%= <- Adresse der Dialogbox
objnull=0
touchexit=&H40
OB_FLAGS(tree%,objnull)=
    OB_FLAGS(tree%,objnull)
    OR touchexit
```

Nun wird bei Anklicken des Grundrahmens, sofern kein anderes Objekt darüber liegt, die *FORM_DO*-Verarbeitung abgebrochen, und man kann in seine selbstdefinierte Verschiebeprozedur verzweigen und nach Erledigung der Bewegung in die Dialogverarbeitung zurückkehren. Um nur das Bewegungszeichen (MOVER) abzufragen, muß man eine Koordinatenüberprüfung, die den zulässigen Bereich einschränkt, durchführen.

Der zweite Unterschied zur Original-AES-Routine liegt darin, daß die modifizierte *FORM_DO*-Funktion auch die Darstellung des Formulares übernimmt. Bei Benutzung der Original-*FORM_DO*-Routine müssen Sie sich um das Zeichnen und Löschen der Dialogbox am Bildschirm selbst kümmern. Hiezu stellt Ihnen das AES die Prozeduren *OBJC_-CENTER, OBJC_DRAW* und *FORM_DIAL* zur Verfügung. Die in GFA modifizierte *FORM_-DO*-Routine nimmt Ihnen diese Arbeit ab. Sowohl die Bildschirmdarstellung als auch das Räumen des Bildschirmes werden automatisch erledigt. Wenn Ihnen dieser Komfort aber hinderlich sein sollte, können Sie diese Programmteile nach Belieben aus der Funktion ausgliedern.

Das Wiederherstellen des Hintergrundes erfolgt im Normalfall durch Absetzen einer REDRAW-Meldung an die laufende Applikation, die nun darauf zu reagieren hat. Falls Sie dies nicht tut, entstehen häßliche graue Löcher, die so manche unsauber programmierte Anwendung auszeichnen. Um das Neuzeichnen der Dialogbox zu beschleunigen, wird in diesem Fall der Bildschirmhintergrund gepuffert. In GFA-BASIC bietet sich für diese Tätigkeit eine Variable vom Typ String an. Der Nachteil liegt darin, daß natürlich mehr Speicherplatz verbraten wird, und zweitens, daß diese Lösung auf der neuen Maschine in der TT-Auflösung bei zu großen Boxen nicht mehr ohne weiteres funktioniert, denn Strings dürfen maximal 32786 Byte groß sein, eine Dimension, die bei einer größeren Box in der TT-Auflösung durchaus überschritten werden kann.

Zur Programmdemonstration wird lediglich eine - mit dem Resource-Construktion-Set erstellte - Dialogbox benötigt. Diese muß allerdings ein Objekt beinhalten, welches als *EXIT* definiert ist. In allen anderen Fällen ist eine Verwendung mit der *FORM_DO*-Routine ein 'Warten auf Reset'. Nach dem Programmstart werden Sie über eine Dateiauswahlbox aufgefordert, eine RSC-Datei zu wählen. Der erste Baum (=Tree1) dieser Datei wird nun am Bildschirm dargestellt und Sie können die 'Fliegenden Dialoge' in GFA-BASIC testen.

Literatur:

[1] Handbuch GFA-BASIC 3.0; GFA Systemtechnik GmbH

[2] Engels G/Görgens M: GFA-BASIC-Version 3.0; GFA Systemtechnik GmbH

[3] Baldauf Mathias: Movedial - Bewegliche Dialogboxen; ST-Computer 1/91, S. 100ff



```
' * * * * * * * * * * * * * * * * * * *
' *
' * FLYING DIAL's in GFA BASIC V3.0
' *
' * Autor: W.Kohlmaier  4/1991
' *
' *(c) 1991  MAXON Computer GmbH
' *
' * * * * * * * * * * * * * * * * * * *
'
' Resourcenamen wählen....
FILESELECT "\*.RSC","",resource$
'
'
' Resource laden
~RSRC_LOAD(resource$)
'
'
baum%=0 ! Adresse von Baum=0 erfragen
~RSRC_GADDR(0,baum%,adresse%)
'
'
'
' ----------- Demonstration einer 'FLY
DIAL' Version in GFA BASIC ------------
'
wx&=32  ! X Koordinate wo Dialogbox
          erscheinen soll
wy&=32  ! Y Koordinate wo Dialogbox
          erscheinen soll
'
'
'
~@form_do(adresse%,0,wx&,wy&) ! die neue
                         FORM_DO Routine
'
'
'
' Resource Speicher wieder freigeben
~RSRC_FREE()
'
END
'
' ------- Funktionen und Proceduren
'
'
FUNCTION form_do(tree%,start%,VAR x&,y&)
  '
  LOCAL form&,d&,w&,h& ! lokale Variablen
  LOCAL rx&,ry&        ! Dummy-Variablen
               um Verschiebung zu testen
  LOCAL puffer$        ! rettet
          Bildschirmausschnitt in Puffer$
```

→

```
LOCAL wrk_x,wrk_y     ! maximale X-,Y-
                       Bildschirmkoordinaten
'
'
wrk_x=WORK_OUT(0)      ! ermittelt über
        VDI die max. Gerätekoordinaten
wrk_y=WORK_OUT(1)
'
sicher_rand=64     ! Sicherheitszone vor
                     dem Hinausschieben
mover_dim=16       ! Größe des MOVER
                     Zeichen
'
' --- Dialogbox zeichnen ---
'
'
~WIND_UPDATE(3)    ! Kontrolle an
                     Dialogbox abgeben
'
'
' Wenn globale X& + Y& Variable
  verwendet werden um die
  Dialogboxkoordinaten
' zu puffern werdfen diese hier auch
  verwendet.
'
IF x&>0 OR y&>0          ! wenn
     Übergabekoordinaten <>0 dann x& y&
     verwenden...
  '
  ~FORM_CENTER(tree%,d&,d&,w&,h&)   ! nur
                        w& und h& erfragen
  OB_X(tree%,0)=x&     ! Dialogbox an x&
  Wert setzen
  OB_Y(tree%,0)=y&     ! Dialogbox an y&
  '
ELSE                     ! ...sonst
                          zentrieren
  '
  ~FORM_CENTER(tree%,x&,y&,w&,h&)
  '
ENDIF
'
' Ausschnitt puffern
GET MAX(0,x&-4),MAX(0,y&-4),MIN(wrk_x,x&+
       w&+4),MIN(wrk_y,y&+h&+4),puffer$
'
'
' Der Trick wieso alles funktioniert...
' Den Basisrahmen der Dialogbox = Objekt
  0 als TOUCHEXIT definieren.
' Aus diesem Grund kann die Box
  verlassen werden wenn die 'MOVE'-Ecke
' angeklickt wird.
'
'
objnull=0
touch_exit=&H40
OB_FLAGS(tree%,objnull)=OB_FLAGS(tree%,
objnull) OR touch_exit  ! Objekt 0 der
Box als TOUCHEXIT definieren
'
'
' ~FORM_DIAL(1,0,0,0,0,x&,y&,w&,h&) !
  eventuell Dialogauschnitt anmelden
'
'
' Dialogbox mit 'MOVE'-Eck zeichnen
form_draw
'
'
' Beginn der eigentlichen FORM_DO Routine
'
'
DO
  '
  ' Dies ist der Standard FORM_DO Aufruf
    des AES
  form&=FORM_DO(tree%,start%)
  '
  '
  ' Schleife verlasssen wenn ein Objekt
    größer als 0 angeklickt wurde...
  ' 'AND &HFF' filtert den Doppelklick
    heraus
  '
  EXIT IF (form& AND &HFF)<>0 ! ändern
  '
  ' wenn das Objekt =0 gewählt wurde
  ' dann Verschiebung beginnen
  '
  '
  rx&=x& ! alte x Koordinaten puffern
  ry&=y& !  -    y    -
  '
  '
  IF MOUSEX<=mover_dim AND MOUSEY<=
     mover_dim  ! wenn Maus innerhalb des
                  MOVE-Zeichens
    IF BTST(form&,31)
      ' Wenn Doppelklick dann Dialogbox
        zentrieren
      '
      ~FORM_CENTER(tree%,x&,y&,w&,h&)
      '
    ELSE
      ' ...sonst Rahmen verschieben
      '
      DEFMOUSE 4 ! Mausform auf flache
                   Hand schalten
      '
      ' Der Wert '22' sperrt die
        Verschiebung in die Menuleiste
      ' kann bei Bedarf auf Null gesetz
        werden
      '
      ' sicher_rand bestimmt die Größe
        des minimal sichtbaren Anteiles der
      ' Dialogbox, bei Verschiebung nach
        rechts unten.
      '
      ~GRAF_DRAGBOX(OB_W(tree%,0),
          OB_H(tree%,0),OB_X(tree%,0),
          OB_Y(tree%,0),4,22,wrk_x+
          OB_W(tree%,0)-sicher_rand,wrk_y+
          OB_H(tree%,0)-sicher_rand,x&,y&)
      '
      DEFMOUSE 0 ! Mausform auf Pfeil
                   retour
      '
    ENDIF
  ENDIF
  '
  '
  '
  IF x&<>rx& OR y&<>ry&
    ' wenn tatsächlich eine Verschiebung
      erfolgte...
    '
    ' alten Ausschnitt restaurieren...
    PUT rx&-4,ry&-4,puffer$
    '
    ' ... und neuen Ausschnitt puffern
    GET MAX(0,x&-4),MAX(0,y&-4),
        MIN(wrk_x,x&+w&+4),
        MIN(wrk_y,y&+h&+4),puffer$
    '
    ' neue Dialogkoordinaten festsetzen
    OB_X(tree%,0)=x&      ! Dialogbox an
                        neuen x& Wert setzen
    OB_Y(tree%,0)=y&      ! Dialogbox an
                        neuen y& Wert setzen
    '
    ' Dialogbox mit 'MOVE'-Eck
      neuzeichnen
    form_draw
    '
  ENDIF
  '
  '
LOOP
'
OB_STATE(tree%,form& AND &HFF)=
OB_STATE(tree%,form& AND &HFF) AND
&X1111111111111110
' --- FORM_DO verlassen  ---
'
'
PUT x&-4,y&-4,puffer$ ! Bildschirm
                        restaurieren
'
'
' ~FORM_DIAL(3,0,0,0,0,x&,y&,w&,h&) !
  ev. Redraw-Meldung ans System schicken
```

→

```
'
'
~WIND_UPDATE(2)  ! Kontrolle ans AES zurückgeben
'
' gibt die Nummer des Objektes, über
  welches die Dialogbox verlassen wurde
' als Rückgabewert zurück
RETURN form&
'
'
ENDFUNC
'
'
PROCEDURE form_draw
  LOCAL m
  '
  ' Diese Procedure zeichnet die Dialogbox
    und das 'MOVE' Eck links oben
  '
  '
  ' Grafik Nullpunkt auf linke, obere Ecke
    der Dialogbox setzen
  CLIP  OFFSET OB_X(tree%,0),OB_Y(tree%,0)
  '
  ' Dialogbox zeichnen
  ~OBJC_DRAW(tree%,0,4,x&-4,y&-4,w&+4,h&+4)
  '
  '
  GRAPHMODE 1      ! Grafikmodus auf
                     überschreiben setzen
  DEFFILL 1,0      ! weißes Füllmuster wählen
  '
  m=mover_dim
  '
  PBOX 0,0,m,m       ! zeichnet den Winkel
                       ins Eck
  DRAW 0,m TO m,0
  '
  SUB m,2
  DRAW 0,m TO m,0
  '
RETURN
```

ARITHMETIK MIT 64-BIT-ZAHLEN

Roman Hodek

DER DATENTYP LONG IN C, ALSO 32-BIT-VARIABLEN, UMFASST EINEN BEREICH VON CA. 4,3 MILLIARDEN ZAHLEN. DAS SIEHT AUF DEN ERSTEN BLICK UNERMESSLICH VIEL AUS, BEI BESTIMMTEN AUFGABENSTELLUNGEN ABER REICHT SELBST DAS NICHT. DESHALB WIRD HIER EIN ROUTINENPAKET ZUM UMGANG MIT 64-BIT-ZAHLEN VORGESTELLT. DIESER TYP DUBLONG HAT EINEN WERTEBEREICH VON CA. 18,5 TRILLIONEN ZAHLEN!

Sie werden sich bestimmt fragen, wie man es eigentlich anstellt, an die Grenzen des long-Typs zu stoßen. Aber es geht schneller, als man denkt. Stellen Sie sich folgende Berechnung vor: Es soll ermittelt werden, wieviel Prozent x Mark von y Mark sind. Um Rundungsfehler zu vermeiden, verwendet man nicht float-Variablen, sondern longs. Da man so keine Nachkommastellen darstellen kann, werden die Beträge in Pfennig angegeben. Das Ergebnis soll auf drei Stellen nach dem Komma genau sein, d.h. ist mit 1000 multipliziert (12.345 % -> 12345). Die Multiplikationen müssen vor der Division erfolgen, damit sich keine Abschneidefehler ergeben. Die Formel zur Berechnung lautet dann:

```
erg = x * 100 * 1000 / y
```

Nehmen wir jetzt als Zahlenbeispiel x = 300 DM, in unserer Darstellung 30000. Der größte Zahlenwert, der in der Berechnung erreicht wird, ist dann:

```
30000 * 100 * 1000 = 3e9
```

Das paßt schon nicht mehr in eine long-Variable, denn die faßt nur Werte bis +2.147e9, es ergibt sich somit ein Überlaufsfehler. Schon bei diesen relativ kleinen Beispielzahlen wird also der Wertebereich von 32 Bit „gesprengt".

```
        Faktor 1                      Faktor 2
[Fakt1 high|Fakt1 low]  x  [Fakt2 high|Fakt2 low]   =      34 x 57 =
-------------------------------------------------------------------
   [Fakt1 high x Fakt2 high]                                  15
+            [Fakt1 high x Fakt2 low]                      +  21
+            [Fakt2 high x Fakt1 low]                      +  20
+                      [Fakt1 low x Fakt2 low]             +  28
-------------------------------------------------------------------
=  [          |          |          |          ]           =  1938
                   64-bit Ergebnis
```

Bild 1: Multiplikation 32x32 -> 64 Bit durch Benutzung des MUL-Befehls

Grundlegende Operationen mit dublongs

Um einen neuen Datentyp *dublong* für 64-Bit-Variablen verwenden zu können, brauchen wir als erstes eine Struktur dafür. Dazu nimmt man einfach 2 longs, wobei aber die Low-Hälfte ein *unsigned long* sein muß, damit Vergleiche richtig funktionieren.

Beachtenswert ist, daß dublong-Strukturen direkt als Parameter übergeben werden können, nicht nur Zeiger auf sie. Die meisten C-Compiler behandeln das so, als würden da getrennt 2 longs stehen. Ihre Reihenfolge ist genauso, wie sie in der Struktur stehen, d.h. zuerst High, dann Low. Im Paket wird das auch ab und zu umgekehrt und 2 longs statt einem dublong als Parameter verwendet. Dadurch kann man sich oft *dl_assign()* sparen.

Nicht ganz so einfach ist es beim Rückgabewert. Ist das Ergebnis einer Funktion ein dublong, kann es nicht direkt mit *return()* zurückgegeben werden. Folglich muß der Funktion zusätzlich ein Zeiger mitgegeben werden, der ihr sagt, wohin das Resultat geschrieben werden soll. Zur Vereinfachung von Rechenausdrükken wird dieser Zeiger (freilich unverändert) als Funktionswert zurückgeliefert, so daß im selben Term mit dem Ergebnis weitergerechnet werden kann. Ein Beispiel wird klarer machen, was ich meine:

```
erg = dl_div( *dl_add( dub1, dub2,
        &dub3 ), dub4, &rest );
```

entspricht (in „normalen" Typen):

```
erg = (n1+n2)/n4;
```

dub3 wird als Platz für das Ergebnis der Addition benötigt. Es muß jedoch von *dl_div()*

nicht direkt angesprochen werden, so daß man die Operationen ineinanderschachteln kann. Das ist so nur möglich, wenn das Resultat von *dl_add()* mit '**dl_add(..)*' angesprochen werden kann.

Als nächsten Schritt bei der Einführung eines neuen Typs muß man Funktionen bereitstellen, die die Umwandlungen zwischen dem neuen und bekannten Typen erledigen. Im dublong-Paket nehmen diese Aufgaben *dl_todl()* und *dl_tolong()* wahr. *dl_todl()* wandelt long in dublong, indem es den long-Wert als Low-Hälfte der dublong betrachtet. Die High-Hälfte ist 0, wenn die long-Zahl größer oder gleich Null ist, sonst -1. Dieser Vorgang, den man Vorzeichenerweiterung nennt, geschieht analog zu den EXT-Befehlen der 68000-Maschinensprache, die Byte auf Wort oder Wort auf Long erweitern. *dl_tolong()* gibt den Wert einer dublong als long zurück. Damit das funktionieren kann, muß der Wert natürlich in eine long-Variable passen, sonst ist das Ergebnis ohne Sinn.

Eine weitere einfache Operation ist das Negieren. Man könnte dafür zwar auch die Subtraktion benutzen (-x = 0-x), aber schneller geht es, wenn man beide Hälften mit dem '~'-Operator NOT-iert, d.h. alle Bits umdreht, und dann 1 addiert. Das ist genau die Konvention, die intern zur Darstellung von negativen Zahlen benutzt wird, das sog. Zweierkomplement.

Addition und Subtraktion

Addition und Subtraktion von dublongs gestalten sich recht einfach, da die benötigten Operationen bereits in der 68000-Maschinensprache enthalten sind. Es gibt neben den einfachen Addier- bzw. Subtrahierbefehlen ADD und SUB Varianten, die den Übertrag einer vorangegangenen Operation berücksichtigen: ADDX und SUBX. Diese dürfen aber nur auf zwei Datenregister angewandt werden.



Bild 2: Prinzip der Multiplikation 64x64 -> 64 Bit durch fortgesetzte Addition (exemplarisch verkürzt auf 8x8 -> 8 Bit)

Zum Addieren zweier dublongs addiert man also zuerst die Low-Hälften mit ADD.L, wobei ein Übertrag entstehen kann. Dann addiert man die High-Hälften mit ADDX.L, wobei der vorherige Übertrag berücksichtigt wird. Das Subtrahieren erfolgt entsprechend. Mit weiteren ADDX/SUBX könnte man auch beliebig lange Zahlen addieren bzw. subtrahieren.

Multiplikation

Etwas kniffliger wird es bei der Multiplikation. Die Funktion *dl_mul()* soll zwei longs zu einem dublong verknüpfen, entspricht also dem Maschinenbefehl MUL in doppelter Breite.

Um unsere Multiplikation durchzuführen, gehen wir vor wie jemand, der nur das kleine Einmaleins kennt und zweistellige Zahlen multiplizieren will (und keinen Taschenrechner besitzt): Man multipliziert die Stellen einzeln miteinander und addiert die Ergebnisse entsprechend ihrer Wertigkeit. Übertragen heißt das: das kleine Einmaleins sind die 16-Bit-Zahlen, deren Produkte mit MUL berechenbar sind. Die Analogie sehen Sie in Bild 1.

Um uns das Leben etwas einfacher zu machen, gehen wir von positiven Faktoren aus. Das heißt, wir ermitteln, welches Vorzeichen das Ergebnis haben wird, und machen die Faktoren einfach positiv. Am Ende der Multiplikation muß dann eventuell das Ergebnis noch negativ gemacht werden.

Um das Addieren der Teilergebnisse etwas cleverer zu gestalten, gehe ich dabei folgendermaßen vor: Zuerst wird *High1 * High2* und *Low1 * Low2* berechnet und daraus ein dublong gebildet. Dann müssen noch *High1 * Low2* und *High2 * Low1* zu der Mitte des Ergebnisses addiert werden. Dabei kann aber jeweils ein Übertrag in das höchstwertigste Wort entstehen. Um diesen zu addieren, holen wir das oberste Wort in ein Register (D1) und addieren mit ADDX den Wert 0. Das bewirkt, daß D1 um 1 erhöht wird, wenn aus der vorangegangenen Addition ein Übertrag entstanden ist, sonst bleibt es unverändert. Da bei ADDX nur Register erlaubt sind, benutze ich D2 als Nullregister. Zum Schluß muß D1 natürlich noch zurückgeschrieben werden.

Da *dl_mul()* nur long mal long rechnen kann, gibt es noch eine zweite Multiplikationsroutine im Paket: *dl_mul2()*. Damit können zwei dublongs verknüpft werden, jedoch hat das Ergebnis nicht immer in einem dublong Platz. Auch z.B. das Resultat von int mal int paßt ja nicht immer in eine int-Variable. Man muß also hier aufpassen, daß man keine zu großen Zahlen nimmt. Das liegt - ganz C-Philosophie - in der Verantwortung des Programmierers.

Doch zur Arbeitsweise von *dl_mul2()*: Hier kann man das ganze leider nicht so einfach auf ein paar MULs zurückführen. Es müßten nämlich sage und schreibe 16 MUL-Befehle sein, dazu kommen noch die Additionen und Übertragsrechnungen. Schneller geht es, indem man die Multiplikation auf Addition und Schieben zurückführt. Das Prinzip ist dasselbe wie oben, nur daß jetzt das Einmaleins recht klein ist: entweder addieren wir den ersten Operator zum Ergebnis oder nicht. Ob das passiert, hängt vom höchstwertigsten Bit des 2. Operators ab. Nach jeder Abfrage schieben wir diesen nach links, so daß das nächste Bit ganz links steht. Zusätzlich nehmen wir das Ergebnis mal 2, sprich schieben es auch nach links, und wiederholen das ganze für alle 64 Bits. Damit entsteht so etwas wie in Bild 2 zu sehen ist. Dort wird es zwar nur für 8 Bit dargestellt, aber so spart man sich lange Zahlenkolonnen, und der Vorgang wird trotzdem klar.

Die Division

Als die schwierigste Aufgabe erweist sich die Division. Sie kann nämlich nicht wie die Multiplikation auf die vorhandenen Maschinenbefehle zu-

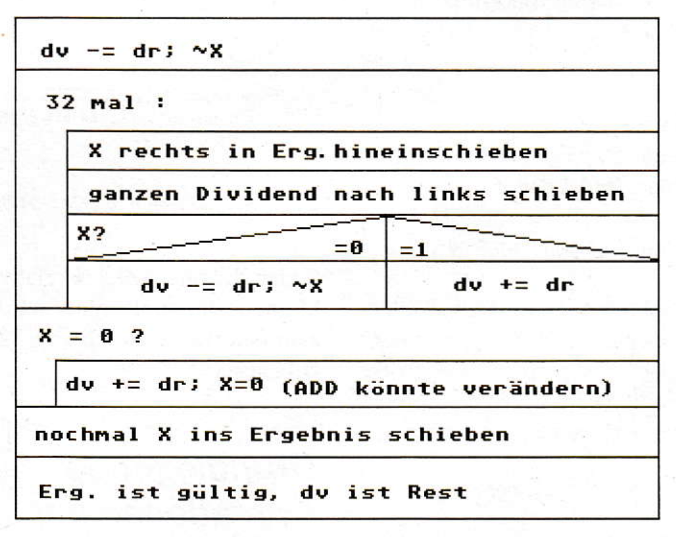


Bild 3: Prinzip der Division als Struktogramm

rückgeführt werden. Es muß also eine Methode eingesetzt werden, die die auf Subtraktionen zurückführt. Die Idee, wie vorher ein ähnliches Prinzip wie die schriftliche Division anzuwenden, erweist sich leider als recht ineffizient.

Die schnellste Lösung ergab ein Algorithmus aus [1], den ich auf 64-Bit-Zahlen und 68000er-Assembler anpaßte. Er arbeitet nach folgendem Schema: Von der High-Hälfte des Dividenden wird der Divisor abgezogen. Nach jeder Subtraktion muß das Übertrags-Flag X negiert werden. Dieses Bit wird dann von rechts in das Ergebnis hineingeschoben, und auch der Dividend wird nach links geshiftet. Abhängig vom X-Flag wird der Divisor entweder addiert oder subtrahiert. Dieser Vorgang wird 32mal wiederholt und zum Schluß - wenn nötig - der Rest positiv gemacht. Das Prinzip dieses Algorithmus' ist in Bild 3 dargestellt.

Das ist - offen gesagt - etwas kompliziert, auch die mathematischen Grundlagen sind nicht ohne weiteres offensichtlich, aber es funktioniert.

Umwandlung ASCII -> dublong

Als kleines Extra ist im dublong-Paket noch eine Funktion *atodl()* enthalten, die ähnlich *atoi()* oder *atol()* einen String in eine dublong-Variable liest. Sie funktioniert nach dem gleichen Algorithmus wie alle diese Routinen: Eine Ziffer wird gelesen und zum bisherigen Wert addiert. Bevor die nächste Zahl betrachtet wird, muß der aktuelle Wert noch mit 10 multipliziert werden. Abgebrochen wird der Vorgang, wenn ein Zeichen kommt, das keine Zahl ist. Falls am Anfang des Strings ein '-' stand, muß das Gesamtergebnis noch negiert werden. Mit Hilfe dieser Prozedur kann man somit Konstanten verwenden, die nicht in longs passen, oder dublongs aus Benutzereingaben lesen.

Literatur:

[1] Zaks, Rodney: Programming the 6502, Sybex Inc., Berkeley 1980



```
/* Headerfile zu 'dublong.c' */

typedef struct {
  long          high;
  unsigned long low;
} dublong;
                                  /* Parameter :     */
extern dublong *dl_todl();        /* ( l, &dl )      */
extern long     dl_tolong();      /* ( dl )          */
extern dublong *dl_neg();         /* ( &dl )         */
extern dublong *dl_add();         /* ( dl,dl,&dl )   */
extern dublong *dl_sub();         /* ( dl,dl,&dl )   */
extern dublong *dl_mul();         /* ( l, l, &dl )   */
extern dublong *dl_mul2();        /* ( dl,dl,&dl )   */
extern long     dl_div();         /* ( dl, l, &l )   */
extern dublong *atodl();          /* ( str, &dl )    */
```

```
/* Implementierung der essentiellen Funktionen*/
/* für 'double longs', d.h. vorzeichenbehaf-  */
/* tete 64-Bit-Zahlen                         */

/* (c) 1991 MAXON Computer                    */
/* by Roman Hodek                             */

/* Funktionsübersicht:                        */
/* dl_todl()   : erweitert long zu dublong    */
/* dl_tolong() : verwandelt dublong in long;  */
/*               wenn der Zahlenbereich von   */
/*               long überschritten ist, ist  */
/*               das Ergebnis ohne Aussage    */
/* dl_neg()    : negiert eine dublong         */
/* dl_add()    : addiert : dl = dl+dl         */
/* dl_sub()    : subtrahiert : dl = dl-dl     */
/* dl_mul()    : multipliziert : dl = l*l     */
/* dl_mul2()   : multipliziert : dl = dl * dl */
/* dl_div()    : dividiert : l = dl/l und     */
/*                           l = dl%l         */
/* atodl()     : liest ein dublong aus einem  */
/*               String                       */

/* Alle Funktionen, die ihr dublong-Ergebnis  */
/* über einen Zeiger ablegen, liefern diesen  */
/* Zeiger auch als Rückgabewert.              */

/* die dublong-Struktur :                     */
typedef struct {
  long high,low;
} dublong;

/* Grenzwerte des Typs dublong :              */
#define DLONG_BIT  64
#define DLONG_MIN  -9.223372037e18
#define DLONG_MAX  9.223372037e18

#ifndef abs
#define abs(x) ( ((x)<0) ? -(x) : (x) )
#endif


/* erweitert l auf 64 Bit und weist es dl zu  */

dublong *dl_todl( l, dl )
  dublong *dl;
  long     l;

{ dl->low = l;
  dl->high = ( l >=0 ) ? 0 : -1;
  return( dl );
}

/* wandelt dublong 'dl' in long */

long dl_tolong( dl )
  dublong dl;

{ return( dl.low ); }

/* negativiert 'op' ( = monadisches Minus )   */

dublong *dl_neg( op )
  dublong *op;

{ dublong *dl_add();

  /* beide Hälften negieren */
  op->low = ~op->low; op->high = ~op->high;
  /* + 1 wegen Zweierkomplement */
  dl_add( 0L, 1L, op->high, op->low, op );
  return( op );
}

/* addiert die dublongs 'op1' und 'op2', Er-  */
/* gebnis nach 'erg'                          */
/*                     *erg = op1 + op2       */

dublong *dl_add( op1, op2, erg )
  dublong op1, op2, *erg;

{ asm { move.l   erg(A6),A0

        move.l   op1+4(A6),D0 ; Low's addieren
        add.l    op2+4(A6),D0
        move.l   D0,4(A0)

        move.l   op1(A6),D0   ; High's mit Über-
        move.l   op2(A6),D1   ; trag addieren
        addx.l   D1,D0
        move.l   D0,(A0)
  }
  return( erg );
}

/* Subtrahiert dublong 'op2' von dublong 'op2'*/
/* Ergebnis nach 'erg'                        */
/*                       *erg = op1 - op2     */

dublong *dl_sub( op1, op2, erg )
```

→

```
    dublong op1, op2, *erg;

{ asm { move.l   erg(A6),A0

        move.l   op1+4(A6),D0 ; Low's subtra-
        sub.l    op2+4(A6),D0 ; hieren
        move.l   D0,4(A0)

        move.l   op1(A6),D0   ; High's mit Über-
        move.l   op2(A6),D1   ; trag subtrahie-
        subx.l   D1,D0        ; ren
        move.l   D0,(A0)
  }
  return( erg );
}

/* Multipliziert zwei longs zu einem dublong */
/*             *erg = op1 * op2             */

dublong *dl_mul( op1, op2, erg )
  long    op1, op2;
  dublong *erg;

{ register int negflag;

  /* negflag zeigt an, ob das Ergebnis         */
  /* negativ ist                               */
  negflag = ( (op1>0) ^ (op2>0) );

  /* beide Faktoren positiv machen             */
  op1 = abs(op1); op2 = abs(op2);

  /* Registerbelegung :                        */
  /* d1 : höchstwertigstes Wort des Ergebnis- */
  /*      ses                                  */
  /* d2 : immer 0, zum addieren des Übertrags */

  asm { move.l    erg(A6),A0
        clr.w     D2

        move.w    op1+2(A6),D0 ; Low 1 *
        mulu      op2+2(A6),D0 ; Low 2
        move.l    D0,4(A0)

        move.w    op1(A6),D0   ; High 1 *
        mulu      op2(A6),D0   ; High 2
        move.l    D0,(A0)

        move.w    (A0),D1      ; höchstes Wort

        move.w    op1+2(A6),D0 ; Low 1 *
        mulu      op2(A6),D0   ; High 2
        add.l     D0,2(A0)     ; addieren
        addx.w    D2,D1        ; Übertrag add.

        move.w    op1(A6),D0   ; High 1 *
        mulu      op2+2(A6),D0 ; Low 2
        add.l     D0,2(A0)     ; addieren
        addx.w    D2,D1        ; Übertrag add.

        move.w    D1,(A0)      ; höchstes Wort
  }                       /* zurückschreiben */

  /* Vorzeichen des Ergebnisses berichtigen   */
  if (negflag) dl_neg( erg );

  return( erg );
}

/* Multipliziert 2 dublong's zu einem dublong */
/* Überschreitet das Ergebnis den Zahlenbe-   */
/* reich, ist das Ergebnis ohne Aussage !     */
/*              *erg = op1 * op2              */

dublong *dl_mul2( op1, op2, erg )
  dublong op1, op2, *erg;

{ register long work1, work2, counter;
  /* 3 Registervariablen reservieren (d5-d7)  */
  register int negflag = 0;

  /* negflag zeigt an, ob das Ergebnis nega-  */
  /* ti ist; beide Operatoren  werden posi-   */
  /* tiv gemacht                              */
  if ( op1.high<0 ) { dl_neg( &op1 );
       negflag ^= 1; }
  if ( op2.high<0 ) { dl_neg( &op2 );
       negflag ^= 1; }

  /* Registerbelegung :                          */
  /* d0/d1 : Multiplikand                        */
  /* d1/d2 : Multiplikator (wird geschoben)      */
  /* d5    : Bitzähler                           */
  /* d6/d7 : Ergebnis                            */

  asm{  move.l op1(A6),D0   ; Register laden
        move.l op1+4(A6),D1
        move.l op2(A6),D2
        move.l op2+4(A6),D3
        clr.l  D6  ; Ergebnis auf 0 initialis.
        clr.l  D7

        moveq  #63,D5  ; 64 Bits mal

loop:   lsl.l  #1,D3   ; Multiplikator schieben
        roxl.l #1,D2
        bcc    no_add
                       ; wenn herausgeschobenes
        add.l  D1,D7   ; Bit = 1, Multiplikand
        addx.l D0,D6   ; zum Eregbnis addieren

no_add: lsl.l  #1,D7   ; Ergebnis schieben ->
        roxl.l #1,D6   ; erg *= 2

        dbf    D5,loop

        movea.l erg(A6),A0 ; Ergebnis ablegen
        move.l  D6,(A0)
        move.l  D7,4(A0)
  }
  /* Vorzeichen des Ergebnisses berichtigen   */
  if (negflag) dl_neg( erg );

  return( erg );
}

/* dividiert dublong 'op1' durch long 'op2'   */
/* Quotient in long 'erg', Rest in 'rest'     */

long dl_div( op1, op2, rest )
  long    op2, *rest;
  dublong op1;

{ register long divisor, save_ccr;
  /* 2 Registervar. reservieren (d6 und d7)   */
  register int  negflag;

  /* Divison durch 0 -> Exception auslösen     */
  if (op2==0) asm{ divu #0,D0 }

  /* negflag zeigt an, ob das Eregbnis nega-  */
  /* tiv wird                                 */
  negflag = ((op1.high>=0) ^ (op2>0));

  /* beide Operanden positiv machen            */
  op2 = abs(op2);
  if ( op1.high<0 ) dl_neg( &op1 );

  /* Registerbelegung :               */
  /* d0/d1 : Dividend                 */
  /* d2    : Bitzähler                */
  /* d3    : Ergebnis                 */
  /* d6    : Zwischenspeicher für CCR */
  /* d7    : Divisor                  */

  asm{  move.l     op1(A6),D0     ; Register laden
        move.l     op1+4(A6),D1
        move.l     op2(A6),divisor
        moveq      #31,D2

        sub.l      divisor,D0   ; Subtrk. vorab
        move       SR,save_ccr
        bchg       #4,save_ccr
loop:
        move       save_ccr,CCR ; X-Flag ins Erg.
        roxl.l     #1,D3        ; schieben
        lsl.l      #1,D1        ; Dividend rot-
        roxl.l     #1,D0        ; tieren

        btst       #4,save_ccr
        beq        _add
        sub.l      divisor,D0   ; wenn X = 1
        move       SR,save_ccr  ; Dvsr von Dvnd
        bchg       #4,save_ccr  ; subtrahieren
```

→

```
         bra       _loop
_add:    add.l     divisor,D0     ; wenn X = 0
         move      SR,save_ccr    ; addieren
_loop:   dbf       D2,loop

         btst      #4,save_ccr    ; wenn X = 0
         bne       no_add         ; den Rest posi-
         add.l     divisor,D0     ; tiv machen
         bclr      #4,save_ccr
no_add:
         move      save_ccr,CCR ; letztes Bit ins
         roxl.l    #1,D3        ; Erg. schieben

         move.l    D3,divisor   ; 'divisor' ist
                                ; jetzt das Ergeb.
         move.l    rest(A6),A0
         move.l    D0,(A0)
   }

   /* Ergebnis mit richtigem Vorzeichen zu-      */
   /* rückgeben                                  */
   return( negflag ? -divisor : divisor );
}
```

```

dublong *atodl( str, erg )
  register char    *str;
  register dublong *erg;

{ register int negflag;

  /* Vorzeichen '-' da ? */
  if (*str=='-') { ++str; negflag = 1; }
  /* Ergebnis auf 0 initialisieren */
  dl_assign( 0L, erg );

  while( *str>='0' && *str<='9' ) {
    dl_add( *dl_mul2( erg->high, erg->low,
                      0L, 10L, erg ),
            0L, (long)(*str-'0'), erg );
    ++str;
  };

  if (negflag) dl_neg( erg );
  return( erg );
}
```

→

SLIDER MIT REALTIME-UPDATE

Markus Hövener

FlyDial hat sie, WEGA sowieso - die Rede ist von Slidern, bei denen, während man sie verschiebt, die Bildschirmdarstellung aktualisiert wird. Dass derartiges in C relativ simpel ist, soll dieser Artikel zeigen.

Grundsätzlich ist zu bedenken, daß mit der unten vorgestellten Routine *DoRealtimeSlide* nicht Slider von GEM-Fenstern verwaltet werden können; wollen Sie animierte Slider haben, so müssen sich diese in einem Dialog befinden, der in der Regel von einem *form_do* überwacht wird.

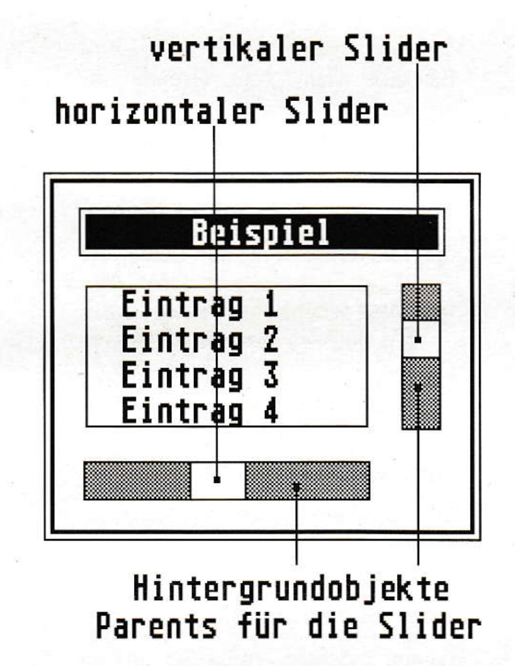
Im RCS

Wie aber muß ein solcher Slider aufgebaut sein? Bild 1 zeigt einen Dialog aus der Sicht des RCS. Man erkennt zwei verschiedene Objekte, die unbedingt benötigt werden: Hintergrund- und Slider-Objekt. Das Hintergrundobjekt ist das Parent für den Slider, der folglich sein Child ist.

Welchen Objekttyp Sie nun im RCS für die jeweiligen Objekte wählen, ist prinzipiell egal. Sie sollten allerdings darauf achten, daß sowohl Hintergrund- als auch Slider-Objekt die gleiche Breite bzw. Höhe haben, und daß Sie beim Hintergrundobjekt das Füllmuster verändern können. Zum Schluß versehen Sie den Slider noch mit dem TOUCHEXIT-Flag, damit das *form_do* abbricht, sobald Sie auf diesen klicken.

Der im Bild zu sehende Abstand zwischen Liste und Slidern sollte im Normalfall nicht vorhanden sein; er ist hier nur dazu da, um die Objektanordnung klarer zu machen.

Bevor wir jedoch zur eigentlichen Routine kommen, folgt noch eine kleine Einführung in ein schönes C-Feature, ohne das *DoRealtimeSlide* nicht möglich wäre...



Funktions-Pointer

Das Problem ist folgendes: Während man den Slider verschiebt, soll irgendetwas am Dialogaufbau geändert werden. Dazu müßte die in Listing 1 beschriebene Routine eine Funktion aufrufen, deren Namen sie aber nicht kennt, also auf dem normale Wege nicht aufrufen kann. Selbstverständlich könnte man in der Routine eigene Funktionen direkt aufrufen; um das Ganze aber universell zu halten, wurde hier der Weg über den Funktions-Pointer gewählt.

Jede Funktion, die Sie in Ihrem Programm deklarieren, steht nach dem Programmstart irgendwo im Speicher. In C erfahren Sie die Stelle, an der die Funktion steht, dadurch, daß Sie den Programmnamen ohne Parameterliste z.B. über ein *printf* ausgeben. Schreiben Sie also *printf(„%p“, Routinenname)*“, gibt das Programm die Adresse der Funktion aus. Selbstverständlich können Sie den Pointer auf die Funktion auch in einer Variablen speichern, wie es im Funktionskopf von *DoRealtimeSlide* im Listing 1 zu sehen ist.

Der Parameter *call_func* zeigt hier auf eine Funktion, die keinen Rückgabewert hat, was man am *VOID* vor *call_func* sieht; außerdem wird der Funktion eine Variable vom Typ WORD übergeben, was hinter *call_func* steht.

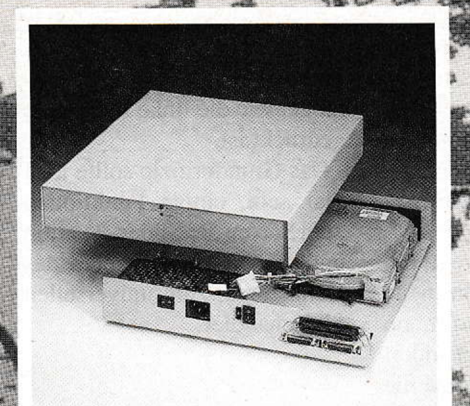
Haben Sie also eine Funktion, die diesen Anforderungen entspricht, können Sie deren Anfangsadresse der Routine *DoRealtimeSlide* übergeben; eine Funktion mit einer anderen Parameterliste eignet sich hierfür nicht. Wie ruft man aber eine Funktion auf, von der man nur die Adresse hat? Die Lösung ist einfach: Der Aufruf erfolgt analog zum normalen Funktionsaufruf. Anstatt des Funktionsnamens schreibt man die Variable, in der man die Adresse der aufzurufenden Funktion gespeichert hat, was in unserem Fall *call_func* ist, und hängt die Parameterliste an. Ein Aufruf könnte also *call_func(1)* lauten.

Vorbereitungen

Bevor man die Routine *DoRealtimeSlide* aufrufen kann, muß man sich die oben erwähnte Funktion schaffen, die dann während des Verschiebens aufgerufen wird. Dieser Funktion wird eine Variable vom Typ WORD übergeben, die die neue

Hochstimmung über Preistief

Westwindzone



Anschlußfertige SCSI-Festplatten für die Atari ST-Serie

- ▶ Datentransferraten > 600 KByte/s, mit Imprimislaufwerken bis zu 1350 KByte/s erzielbar, Zugriffszeiten bis zu 14 ms.
- ▶ Spitzensoftware: 255 Partitionen installierbar, Softwareschreibschutz, jede Partition autobootfähig, Interleave 1:1 einstellbar, Cache, Backup, Optimizer in der Software enthalten.
- ▶ 100% Atari-kompatibel, Fremdbetriebssysteme (PC-Speed, AT-Speed, PC-Ditto, Spectre, Aladin, Minix, OS-9, RTOS) sind voll lauffähig.
- ▶ Sehr leise, 3.5″-Festplatten ohne Lüfter, 5.25″-Festplatten mit gesteuertem Lüfter.
- ▶ Durchgeschleifter gepufferter DMA-Bus, Autoparkfunktion hardwaremäßig, DMA-Nr. von außen einstellbar.
- ▶ Herausgeführter SCSI-Bus, durch abschaltbaren Hostadapter optimale Datenübertragung auch an Apple McIntosh und PCs.
- ▶ Zweite SCSI-Festplatte im Gehäuse nachrüstbar, SCSI-Hostadapter und Gehäuse für interne zweite Festplatte vorbereitet.

Seagate ST157N-1, 49 MB, 28 ms	948,-
Quantum LPS52S, 52 MB, 17 ms	998,-
Seagate ST296N, 85 MB, 28 ms	1.148,-
Seagate ST1096N, 81 MB, 24 ms	1.198,-
Quantum LPS105S, 105 MB, 17 ms	1.298,-
Seagate ST1239N, 200 MB, 15 ms	1.998,-
Imprimis ST2383N, 337 MB, 14 ms	3.598,-
Imprimis ST4766N, 676 MB, 15 ms	5.998,-
Imprimis ST41200N, 1050 MB, 15 ms	9.998,-
Syquest SQ555, 44 MB, 25 ms	1.398,-

SCSI-Kits

Unsere SCSI-Einbaukits bestehen aus Festplattenlaufwerk, SCSI-Hostadapter, DMA-Kabel, SCSI-Kabel, Installationssoftware und Handbuch.

Seagate ST157N-1, 49 MB, 28 ms, Kit	748,-
Quantum LPS52S, 52 MB, 17 ms, Kit	798,-
Seagate ST296N, 85 MB, 28 ms, Kit	948,-
Seagate ST1096N, 81 MB, 24 ms, Kit	998,-
Quantum LPS105S, 105 MB, 17 ms, Kit	1.098,-
Seagate ST1239N, 200 MB, 15 ms, Kit	1.798,-
Syquest SQ555, 44 MB, 25 ms, Kit	1.198,-

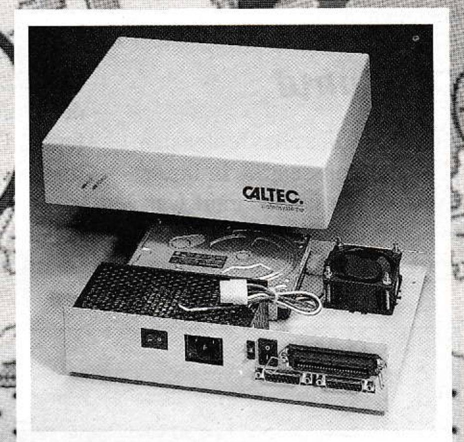
Nachfolgend diverse für SCSI-Festplattenlösungen benötigte Einzelteile:

SCSI-Komplettkit bestehend aus Gehäuse, Netzteil, Hostadapter, Kabel, Software, Handbuch, Montagematerial	398,-
SCSI-Hostadapter incl. Software, Kabel	198,-
Syquest-Cartridge SQ400	198,-
Gehäuse für HDD	99,-
Netzteil 50 Watt	99,-
DMA-Kabel	39,-
SCSI-Kabel	39,-



Datensysteme

Eugenstraße 28
7302 Ostfildern 4
Telefon 07 11 / 4 57 96 23
Telefax 07 11 / 4 56 95 66



Anschlußfertige SCSI-Festplatten für Atari TT und Mega STE

- ▶ Vom Design passend zur Haupteinheit des TT bzw Mega STE.
- ▶ Bei Anschluß an TT können unsere Geräte optional ohne SCSI-Hostadapter betrieben werden, bei Kauf einer anschlußfertigen Festplatte ohne SCSI-Hostadapter reduziert sich der Kaufpreis um DM 150,–
- ▶ mit gesteuertem, sehr leisem Lüfter
- ▶ weitere technische Daten wie unsere SCSI-Festplatten für die Atari ST-Serie.

Seagate ST157N-1, 49 MB, 28 ms	948,-
Quantum LPS52S, 52 MB, 17 ms	998,-
Seagate ST296N, 85 MB, 28 ms	1.148,-
Seagate ST1096N, 81 MB, 24 ms	1.198,-
Quantum LPS105S, 105 MB, 17 ms	1.298,-
Seagate ST1239N, 200 MB, 15 ms	1.998,-
Imprimis ST2383N, 337 MB, 14 ms	3.598,-
Imprimis ST4766N, 676 MB, 15 ms	5.998,-
Imprimis ST41200N, 1050 MB, 15 ms	9.998,-
Syquest SQ555, 44 MB, 25 ms	1.398,-



Slider-Position angibt; bei vertikalen Slidern ist dies die neue y-Position, bei horizontalen die geänderte x-Position des Sliders. Diese Funktion wird aufgerufen, wenn der Slider seine Position geändert hat. Diesem Wert entsprechend können Sie dann den Dialoginhalt verändern, also z.B. den Inhalt einer Liste nach oben oder unten verschieben.

Der Aufruf

Haben Sie dies alles hinter sich gebracht, müssen Sie in Ihrem *form_do* nur noch darauf warten, daß der Slider angeklickt wird. Ist dieses Ereignis eingetreten, können Sie die Funktion *DoRealtimeSlide* aufrufen, der Sie folgende Parameter übergeben:

- den Pointer auf den Objektbaum, in dem sich die Objekte befinden
- den Index des Hintergrund- und des Slider-Objektes
- den Slider-Typ; bei horizontalen Slidern übergibt man HORT_SLIDER, bei vertikalen VERT_SLIDER (siehe Listing 2)
- den Pointer auf die aufzurufende Funktion. Soll keine Funktion aufgerufen werden, müssen Sie hier NULL übergeben, was in STDDEF.H definiert ist.

Der Kern der Routine ist ein *evnt_multi*, der auf zwei verschiedene Ereignisse wartet, nämlich das Loslassen der linken Maustaste und eine Bewegung der Maus.

Wurde die Maus bewegt, berechnet die Routine die neue Slider-Position und prüft, ob sich überhaupt etwas geändert hat. Ist dies der Fall, wird die via *call_func* übergebene Funktion aufgerufen, der die neue Slider-Position übergeben wird. Bevor jetzt der Slider an der neuen Position gemalt wird, muß noch ein Teil des Hintergrundobjektes gezeichnet werden, damit keine Reste vom Slider übrig bleiben. Dieses Geschehen wiederholt sich solange, wie die linke Maustaste gedrückt ist.

Das Grundprinzip sollte jetzt klar sein, einige Funktionen zum Setzen der Slider-Position und -größe komplettieren das Ganze dann zum universellen Sliderhandler.

Literatur:

[1] Jankowski, Reschke, Rabich, ATARI ST Profibuch
[2] Uwe Hax, Oliver Scholz, CPX - Teil 2, ST-Computer 4/91



```
# include <aes.h>
# include <portab.h>


/* Konstanten für slider_type */
#define VERT_SLIDER   1
#define HORT_SLIDER   !VERT_SLIDER


VOID DoRealtimeSlide( OBJECT  *dialptr,
                      WORD back_index,
                      WORD slider_index,
                      WORD slider_type,
                      VOID (*call_func)
                      (WORD new_pos) );
```

```
/*****************************************/
/*    Modul  : REALTIME.C                */
/* Aufgabe  : Slider in Echtzeit         */
/*    Autor  : Markus Hövener            */
/*    Datum  : 7.9.1991                  */
/*          (c) MAXON Computer GmbH      */
/*****************************************/


# include <aes.h>
# include <portab.h>

# include "realtime.h"



/*****************************************/
/* Slider in Echtzeit bewegen            */
/*****************************************/
VOID DoRealtimeSlide( OBJECT  *dialptr,
           WORD     back_index,
           WORD     slider_index,
           WORD     slider_type,
           VOID     (*call_func)
           (WORD    new_pos) )
{
  WORD    /* Aktuelle Mauskoordinaten      */
          m_x, m_y,
          /* Anfängliche Mauskoordinaten   */
          beg_mx, beg_my,
          /* Anfängliche Sliderpositionen */
          dial_x, dial_y,
          /* Koordinaten des 'back_index' */
          abs_x, abs_y,
          /* Maximale Sliderposition */
          max_pos,
          /* Position zum Vergleich   */
          prev_x, prev_y,
          events,
          _void;


  /* Maus als Hand */
  graf_mouse( FLAT_HAND, 0L );


  /* Mauskoordinaten holen */
  graf_mkstate( &m_x, &m_y, &_void,
                &_void );
  beg_mx = m_x;
  beg_my = m_y;


  /* Sliderposition ermitteln */
  dial_x = dialptr[slider_index].ob_x;
  dial_y = dialptr[slider_index].ob_y;


  /* Maximale Verschiebung */
  if( slider_type == VERT_SLIDER )
    max_pos = dialptr[back_index].ob_height
            - dialptr
              [slider_index].ob_height;
  else
    max_pos = dialptr[back_index].ob_width
            - dialptr
              [slider_index].ob_width;


  /* Absolute Koordinaten vom
     Hintergrundobjekt */
  objc_offset( dialptr, back_index, &abs_x,
               &abs_y );


  do
  {
    /* Sliderposition merken */
    prev_x = dialptr[slider_index].ob_x;
    prev_y = dialptr[slider_index].ob_y;


    /* Event */
    events = evnt_multi( /* Maus- und
                Rechteckevents */
                MU_BUTTON|MU_M1,
                /* Mausklick */
                1, 1, 0,
                /* Das Rechteck */
                1, m_x, m_y, 1, 1,
                0, 0, 0, 0, 0,
                /* Message-Buffer
                */
                0L,
                /* Kein Timer */
                0, 0,
                /* Endparameter */
                &m_x,   &m_y,
                &_void,
                &_void, &_void,
                &_void );

```

```
    /* Neue Position errechnen */
    if( slider_type == VERT_SLIDER )
      dialptr[slider_index].ob_y = dial_y +
                                   m_y - beg_my;
    else
      dialptr[slider_index].ob_x = dial_x +
                                   m_x - beg_mx;


    /* Eingrenzen */
    if( slider_type == VERT_SLIDER )
    {
      dialptr[slider_index].ob_y =
      (dialptr[slider_index].ob_y < 0)
      ? 0
      : dialptr[slider_index].ob_y;
      dialptr[slider_index].ob_y =
      (dialptr[slider_index].ob_y >
      max_pos)
      ? max_pos
      : dialptr[slider_index].ob_y;
    }
    else
    {
      dialptr[slider_index].ob_x =
      (dialptr[slider_index].ob_x < 0)
      ? 0
      : dialptr[slider_index].ob_x;
      dialptr[slider_index].ob_x =
      (dialptr[slider_index].ob_x >
      max_pos)
      ? max_pos
      : dialptr[slider_index].ob_x;
    }


    /*************************************/
    /* Änderung                          */
    /*************************************/
    if( (dialptr[slider_index].ob_y !=
         prev_y) ||
        (dialptr[slider_index].ob_x !=
         prev_x) )
    {
      /* Funktion evtl. aufrufen */
      if( call_func )
        call_func(
        dialptr[slider_index].ob_y );


      /* Den Hintergrund neumalen */
      if( slider_type == VERT_SLIDER )
      {
        if( dialptr[slider_index].ob_y >
            prev_y )
          objc_draw( dialptr, back_index,
                     1, abs_x,
                     abs_y + prev_y - 1,
                     dialptr
                     [back_index].ob_width,
                     dialptr[slider_index].
                     ob_y - prev_y );
        else
          objc_draw( dialptr, back_index,
                     1, abs_x,
                     abs_y +
                     dialptr[slider_index].
                     ob_y +
                     dialptr[slider_index].
                     ob_height,
                     dialptr[back_index].
                     ob_width,
                     prev_y -
                     dialptr[slider_index].
                     ob_y + 1 );
      }
      else
      {
        if( dialptr[slider_index].ob_x >
            prev_x )
          objc_draw( dialptr, back_index,
                     1, abs_x + prev_x - 1,
                     abs_y,
                     dialptr[slider_index].
                     ob_x - prev_x,
                     dialptr[back_index].
                     ob_height );
        else
          objc_draw( dialptr, back_index,
                     1, abs_x +
                     dialptr[slider_index].
                     ob_x +
                     dialptr[slider_index].
                     ob_width,
                     abs_y,
                     prev_x -
                     dialptr[slider_index].
                     ob_x + 1,
                     dialptr[back_index].
                     ob_height );
      }

      /* ... und den Slider neumalen */
      objc_draw( dialptr, slider_index, 1,
                 abs_x,
                 abs_y +
                 dialptr[slider_index].ob_y
                 - 1,
                 abs_x +
                 dialptr[slider_index].
                 ob_width,
                 abs_y +
                 dialptr[slider_index].ob_y
                 + dialptr[slider_index].
                 ob_height );
    }
  }
  while( !(events & MU_BUTTON) );


  /* Normale Maus */
  graf_mouse( ARROW, 0L );
```









CONTROL-SHIFT-MAUS

Ralf Stachs

Das Desktop bietet bisher die Möglichkeit, mehrere Dateien oder Icons zu selektieren. Dabei muss man bei gedrückter Shift-Taste die Dateien mit dem Mauszeiger anklicken. Ab TOS 1.4 kann man Dateien auch verschieben, das heisst, die Quelldatei wird kopiert und anschliessend gelöscht. Sollen Dateien verschoben werden, muss die Control-Taste gedrückt sein. Wollte man bisher mehrere Dateien verschieben, musste man die Control- und Shift-Taste drücken und die Dateien mit dem Mauszeiger selektieren. Das Programm Control-Shift-Maus (CS-Maus) ermöglicht jetzt mit Hilfe der rechten Maustaste ein Verschieben oder Kopieren von einzelnen oder mehreren Dateien ohne zusätzliches Drücken der Shift-oder Control-Taste.

Sollen mit CS-Maus mehrere Dateien ausgewählt werden, muß man anstelle der Shift-Taste die rechte Maustaste drükken. Beim Verschieben von Dateien wird auch die Control-Taste durch die rechte Maustaste ersetzt. Das Selektieren der Datei geschieht wie bisher mit einem Klick der linken Maustaste. Will man mit CS-Maus mehrere Dateien verschieben, drückt man einfach die rechte Maustaste wählt die Dateien aus und verschiebt sie bei gedrückter rechter Maustaste in das Zielverzeichnis. Das Selektieren oder Verschieben mehrerer Dateien ist wie bisher mit der Shift-oder Control-Taste möglich. Auch das Kopieren von einem nicht aktiven Fenster in das aktive bei gedrückter rechter Maustaste ist wie bisher möglich.

CS-Maus kann man aus dem Auto-Ordner starten oder direkt vom Desktop. Will man CS-Maus ausschalten, muß das Programm ein zweites Mal gestartet werden. Es wird dann die Meldung 'CS-Maus AUS' ausgegeben. Das Einschalten geschieht durch ein erneutes Starten des Programms. CS-Maus ist nur im Desktop aktiv; sobald ein Programm gestartet wird, schaltet es sich selber aus.

Installation

Nun zur Beschreibung des Listings. Das Programm springt zuerst zur Installierungsroutine (INSTALL). Dort werden eine neue Mausroutine installiert und die Adressen von *kbshift* und *_run* geholt. Die Routine *Kbdvbase* (XBIOS 34) liefert einen Zeiger auf die KBDVECS-Struktur. In dieser Struktur ist der Mausvektor (*mousevec*) enthalten. Das Betriebssystem springt bei jeder Mausbewegung durch diesen Vektor. Da CS-Maus die neue Mausroutine mit dem XBRA-Protokoll anmeldet, kann es auch feststellen, ob CS-Maus schon installiert ist. Eine Verfolgung der Kette ist nur dann möglich, wenn sich jede neue Mausroutine an das XBRA-Protokoll hält. War CS-Maus schon installiert, wird das Programm ein- oder ausgeschaltet. Dazu wird das Flag *MAUS_-OFF* invertiert. Dieses befindet sich 13 Bytes hinter der schon installierten Mausroutine. War die Installation erfolgreich, werden die neue Mausroutine (*NEW_MOUSE*) und die VBL-Routine (*NEU_VBL*) resident im Speicher gehalten.

Aufbau eines Mauspaketes
1. Byte Header von $F8 bis $FB *
2. Byte Mausbewegung in x-Richtung
3. Byte Mausbewegung in y-Richtung

* wird von CS-Maus benutzt

Neue Mausroutine

Wird CS-Maus zum ersten Mal installiert, muß zwischen einem Start aus dem Auto-Ordner und einem vom Desktop unterschieden werden. Deshalb wird versucht, eine Applikation anzumelden (APPL_INIT AES 10). Wird CS-Maus aus dem Auto-Ordner gestartet, ist die *ap_-version* gleich Null, da das AES noch nicht initialisiert ist. Bei einem Start vom Desktop wird der alte Mausvektor in *OLD_-VEC* gespeichert und die Adresse der neuen Mausroutine in die KBDVECS-Struktur eingetragen. Da der Mausvektor erst nach dem Starten aller Programme aus dem Auto-Ordner belegt wird, muß man bei einem Start von CS-Maus aus dem Auto-Ordner anders vorgehen. Dabei wird eine VBL-Routine (*NEU_VBL*) installiert. Diese wartet jetzt solange, bis der Mausvektor überschrieben wird. Ist dies der Fall, wird wie beim Starten vom Desktop der

alte Mausvektor in *OLD_VEC* gespeichert und die Adresse der neuen Mausroutine in die KBDVECS-Struktur eingetragen. Danach wird die VBL-Routine gelöscht.

_run und kbshift

Die Zeiger *_run* und *kbshift* sind in der SYSHDR-Struktur abgelegt. Das ist aber erst ab TOS 1.2 der Fall. Bei älteren TOS-Versionen liegen *_run* bei $602C und *kbshift* bei $E1B. Den Zeiger auf die SYSHDR-Struktur erhält man durch die Systemvariable *_sysbase* ($4-f2). *_run* enthält die Adresse des Zeigers der aktuellen Basepage. Die Länge des Programmcodes ist mit einem Offset von $C in der Basepage abgelegt. Ist die Länge des Programmcodes gleich Null, ist das aktuelle Programm das Desktop. *_run* wird dazu benutzt, CS-Maus auszuschalten, wenn ein Programm gestartet wird. Der Zeiger *kbshift* zeigt auf ein Byte, in dem der Tastaturstatus abgelegt ist. Dabei entspricht Bit 0 der rechten Shift-Taste und Bit 2 der Control-Taste. Man kann den Tastaturstatus auslesen oder setzen.

Mauspakete

Der Tastaturprozessor schickt bei jeder Mausbewegung Pakete zur Mausroutine. Diese Mauspakete bestehen aus drei Bytes, deren Adresse ist im Register A0 enthalten. Das erste Byte enthält den Header. In diesem Byte ist der Status der Mausknöpfe abgelegt. Dabei entspricht Bit 0 der rechten und Bit 1 der linken Maustaste. Ist eine Taste gedrückt, ist das entsprechende Bit gesetzt. Der Wert des Headers liegt bei Mausbewegungen von $F8 bis $FB. Die relative Mausbewegung in x-Richtung enthält das zweite Byte. Die Mausbewegung in y-Richtung ist im dritten Byte enthalten. Die neue Mausroutine, die nach der erfolgreichen Installation jetzt jedes Mauspaket zuerst erhält, benötigt nur das erste Byte.

Shift und Control

Die neue Mausroutine testet zuerst, ob ein Mauspaket angekommen ist; danach, ob CS-Maus ein- oder ausgeschaltet ist. Wenn der Programmcode gleich Null ist (Desktop aktiv), wird der Maus-Header auf einen rechten Mausdruck getestet. Ist dies der Fall, wird der Zeiger auf den Tastaturstatus geholt. Nun setzt man Bit 0 (Shift) und Bit 2 (Control). Die beiden Bits werden gelöscht, sobald eine Mausbewegung erfolgt, bei der nicht die rechte Maustaste gedrückt ist. Auch wenn das aktuelle Programm nicht mehr das Desktop ist, werden die Bits gelöscht.

Literatur:

[1] Atari ST Profibuch, Sybex
[2] ST Computer 6/90, Seite 83, Springmaus



Bit-Belegung des Tastaturstatus'
Bit 0: Shift-Taste rechts *
Bit 1: Shift-Taste links
Bit 2: Control-Taste *
Bit 3: Alternate-Taste
Bit 4: Caps Lock gesetzt
Bit 5: Maustaste rechts (Clr Home)
Bit 6: Maustaste links (Insert)
Bit 7: reserviert

* wird von CS-Maus benutzt

Offset	Bezeichnung	Beschreibung
0	midivec	Midi-Eingabe
4	vkbderr	Tastaturfehler
8	vmiderr	Midi-Fehler
12	statvec	Status von IKBD lesen
16	mousevec	Mausabfrage *
20	clockvec	Uhrzeitabfrage
24	joyvec	Joystick-Abfrage
28	midisys	Midi-Systemvektor
32	ikbdsys |	KBD-Systemvektor

Belegung der KBDVECS-Struktur

Offset	Bezeichnung	Beschreibung
0	os_entry	Einsprungadresse
2	os_version	TOS-Version *
4	os_start	Startadresse des OS-Codes
8	os_base	Anfangsadresse des Betriebssystems
12	os_membot	Beginn des freien RAM
16	os_shell	Default Shell
20	os_magic	Zeiger auf GEM-Magic-Variable
24	os_gendat	Erstellungsdatum BCD-Format
28	os_palmode	PAL oder NTSC
30	os_gendatg	Erstellungsdatum GEMDOS-Format
Folgende Variablen erst ab TOS 1.2		
32	*_root	Zeiger auf 'mifl'-Liste
36	kbshift	Zeiger auf Bit-Belegung des Tastaturstatus *
40	**_run	Zeiger auf aktuelle Basepage *

Belegung der SYSHDR-Struktur

```
*-------------------------*
*  Control-Shift-Maus     *
*  by Ralf Stachs         *
*-------------------------*
* (c) 1991 MAXON Computer *
*-------------------------*

;TRAPS
GEMDOS  equ 1
BIOS      equ 13
XBIOS     equ 14


*******************************************
;Programmteil der resident im Speicher
;gehalten wird v. RESI_ANFANG bis RESI_ENDE
*******************************************
    TEXT
RESI_ANFANG:
        bra INSTALL
            ;zur Installationroutine springen

;Flags
******
    ;FIRST       ->Rechte Maustaste zum
                   ersten mal gedrückt
    ;            0->Rechte Maustaste noch
                   nicht gedrückt
    ;            1->Rechte Maustaste war
                   schon gedrückt
    ;MAUS_OFF    ->schaltet CS_Maus an oder
                   aus
    ;            $00 CS_Maus an
    ;            $FF CS_MAUS aus

FIRST:         dc.b 0
MAUS_OFF:      dc.b 0

          dc.b "XBRA"   ;XBRA Protokoll
          dc.b "RS11"   ;eigene Kennung
OLD_VEC:       dc.l 0   ;alter Mausvektor

```

→

PREISSENKUNGEN BEACHTEN!!!

HBS 240 nur 299 DM

Das Speederboard der Spitzenklasse mit 16 MHz und 16 KB Cache sowie optional mit FPU 68.881-16! Lesen Sie dazu den Testbericht im ST-Magazin 11/91: "STÄRKEN: solide Bauart, reichlicher Geschwindigkeitszuwachs, viele zusätzlichen Features, günstiger Preis", "FAZIT: eine echte Alternative auf dem Beschleunigerkarten-Markt"

RAMCARD 3 MB 255 DM

RAMCARD die preiswerte Lösung für jeden, der weiß an welchem Ende der Lötkolben heiß wird! Der Profi weiß, die beste Verbindung ist die gelötete Verbindung! Für STs mit 1 MB.

FESTplatten:

anschlußfertig, temperaturgeregelt, DMA-Bus, SCSI-Bus, Echtzeituhr!!!

HD-52 MB Quantum	1044.-
HD-105 MB Quantum	1398.-
HD-210 MB Quantum	2111.-
HD-330 MB Imprimis	3099.-
HD-530 MB Conner	3999.-
HD-670 MB Imprimis	4444.-

Wechselplatten inklusive 44 MB Medium:

Syquest 44 MB	1255.-
+ 52 MB HD	1888.-
+ 105 MB HD	2333.-
+ 210 MB HD	2999.-

zusätzliche Medien 44 MB:

1-1	2-4	ab 5
169.-	159.-	152.5

Quantum 52 MB 17mS	555.-
Quantum 105 MB 17mS	888.-
Quantum 210 MB 15mS	1777.-
Conner 530 MB 15mS	2999.-

IMEX 3 MB 277 DM

IMEX 4 MB 399 DM

IMEX, Speicher für alle STs

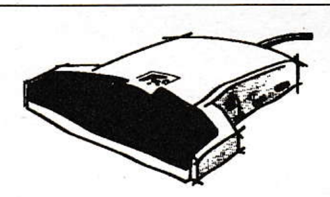
HBS 240 16MHz, 16KB Cache	299.-
68.881-16 für HBS 240	222.-
HBS 240 inkl. 68.881	499.-
HBS 110 16MHz, 0KB Cache	111.-
Mega STE Coprozessor	99.-
RAM-Erweiterungen	
1040 / Mega STE 2 MB	222.-
1040 / Mega STE 4 MB	422.-
RAMCARD 3 MB	255.-
IMEX 3 MB	277.-
IMEX 4 MB	399.-
IMEX upgrade 3 auf 4 MB	177.-
AT-Speed C16 16 MHz	499.-
Autoswitch OverScan	111.-
NVDI	99.-
OverScan + NVDI	199.-
MegaScreen	222.-
Reflex 1024	799.-
HD-Laufwerk Teac	111.-
HD-Kit intern	111.-
silent Lüfter	55.-
NEU! NEU! NEU! PureC	377.-
TEX auf 11 Disketten	33.-
Paket mit FONTS und Utilities für Signum + Script 7 Disketten	22.-
Arabesque	249.-
Arabesque Pro	333.-
CyPress	255.-
Interface (RSC-Editor)	88.-
THEMADAT (Datenbank)	222.-
1ST fibuMAN	149.-
fibuMAN e	388.-
fibuMAN f	666.-
fibuMAN m	888.-
fibuSTAT	333.-
Disketten 3.5" 50 Stück	44.-
51.1000-80/70/60	9.5
27C256-120 6 Stück	39.-
27C512-120 6 Stück	66.-
Mega ST1 1 MB	888.-
Mega ST1 3 MB	1222.-
Mega ST1 4 MB	1333.-
Mega ST1 1 MB + HBS 240	1222.-
Mega ST1 3 MB + HBS 240	1555.-
Mega ST1 4 MB + HBS 240	1666.-
SM 124	249.-
9 Nadel Printer SP-1900	366.-
9 Nadel Printer SP-2000	444.-
24 Nadel Printer SL-92	666.-
LapTopPrinter LT-20	888.-
HP deskjet 500	999.-

Friedhelm Heyer und David Neumann GbR Hardwareentwicklungen - Promenadenstr. 50 - W-5100 Aachen - Bürozeiten: Mo-Fr 10-13 und 14-17 Uhr - Tel: 0241/35247 - FAX: 0241/35246
Zwischenverkauf vorbehalten! Versandkosten: Vorkasse 10 DM, Nachnahme 15 DM (wahlweise Post oder UPS) - Bankverbindung: Postgiroamt Köln, BLZ 370 100 50 KTN 66 937-505

---

edv komplett GmbH

König-Karl-Str. 49
7000 Stuttgart 50
Telefon 07 11/55 77 82
Fax 07 11/55 77 83
Btx 07 11/55 77 84



Scanner

Logitech Scanner mit Repro Studio Junior	548,00
Logitech Scanner mit Repro Studio Junior und Avant trace	698,00



Drucker Sonderpreise

Seikosha SP-1900	378,00
Seikosha SL-92 24 Nadel	648,00

Hardware und Zubehör

Tower für TT/Mega STE	398,00
SCSI Festplatte 42MB	998,00
Wechselplatte 44MB	1598,00
Großbildschirme	a.A.
That's Mouse	78,00
boeder Maus	49,80

PC Emulatoren

AT Speed	348,00
AT Speed C16	498,00

Textverarbeitung

That's Write 2.0	378,00
1st Word Plus 3.15	148,00

DTP

Calamus 1.09	548,00

Sonstiges

alle Prospero-Programme	a.A.
alle Omikron-Programme	a.A.
PD-Programme	8,00

Fordern Sie unseren Gesamtprospekt oder Einzelprospekte über Produkte, für die Sie sich interessieren, an!

Wir führen auch hier nicht aufgeführte Produkte rund um den Atari!

Infoanforderung

Ich/wir möchten gerne weitere Informationen über die von Ihnen vertriebenen Produkte. Bitte senden Sie mir:

Gesamtkatalog ja nein

Info zu: ____________

Meine Adresse:

Name: ____________

Straße: ____________

PLZ: _____ Ort: ____________

Telefon: ____________

```
;neue Mausroutine
;****************
NEW_MOUSE:
        movem.l d0/a0-a2,-(sp)  ;Register retten
;Maus Header?
        move.b (a0),d0   ;Header Kopf in d0
        cmp.b #$f8,d0    ;kleiner $f8
        blt ENDE         ;ja
        cmp.b #$fb,d0    ;größer $fb
        bgt ENDE         ;ja
;Maus ausgeschaltet TOS?
        move.l A_MOUSE_FLAG,a1  ;Flag für Maus aus (MOUSE_FLAG)
        tst.b (a1)       ;Maus ausgeschaltet
        bne ENDE         ;ja
;CS_Maus ausgeschaltet?
        tst.b MAUS_OFF   ;CS_MAUS ausgeschaltet
        bne ENDE         ;ja
;aktuelles PRG Desktop?
        move.l A_RUN,a1  ;a1=Adresse auf Zeiger der aktuellen Basepage
        move.l (a1),a2   ;a2=Zeiger auf Basepage
        tst.l $C(a2)     ;Länge des Programmcode = 0
        bne SP_11        ;nein, nicht Desktop
;Rechte Maustaste gedrückt
        btst.b #0,(a0)   ;rechte Maustaste gedrückt
        beq SP_11        ;nein, Flags und Tasten löschen
                         ;wenn FIRST noch gesetzt
;Rechte Maustaste zum ersten mal gedrückt
        tst.b FIRST      ;zum ersten mal
        bne ENDE         ;nein
;Control und Shift Taste setzen
        move.l A_KBSHIFT,a1  ;Adresse von Kbshift in a1
        or.b #%00000101,(a1) ;Control Taste (Bit 2) setzen
                             ;und Shift Taste (Bit 0) setzen
        move.b #1,FIRST      ;Flag setzen
        bra ENDE
;Control und Shift Taste löschen
SP_11:  tst.b FIRST          ;First Flag noch gesetzt
        beq ENDE             ;Nein, dann Ende
        move.l A_KBSHIFT,a1  ;Adresse von Kbshift in a1
        and.b #%11111010,(a1) ;Control-Taste (Bit 2) löschen
                             ;und Shift Taste (Bit 0) löschen
        move.b #0,FIRST      ;Flag löschen
;alte Mausroutine anspringen
ENDE: movem.l (sp)+,d0/a0-a2  ;Register zurückschreiben
        move.l OLD_VEC,-(sp) ;alte Mausroutine
        rts                  ;anspringen
;VBL-Routine bei Start aus AUTO-Ordner
;***************************************
        dc.b "XBRA"          ;XBRA Protokoll
        dc.b "RS12"          ;eigene Kennung
        dc.l 0               ;alter vektor nicht benutzt
NEU_VBL:
        move.l A_KBDV,a0     ;kdbv Adresse nach a0
        lea 16(a0),a1        ;Adresse Mausvektor a1
        lea OLD_VEC,a0       ;alter Mausvektor vom Betriebssystem
        cmp.l (a0)+,(a1)+    ;alter Mausvektor noch da
        beq SP_3             ;ja
        move.l -(a1),OLD_VEC ;alten Mausvektor sichern
        move.l #NEW_MOUSE,(a1)  ;neue Mausroutine initialisieren
        move.l A_VBL_SLOT,a0 ;vbl-Slot löschen
        move.l #0,(a0)
SP_3: rts
;Adressen
;********
    ;A_KBDV        -> Adresse der kbdv Tabelle
    ;A_VBL_SLOT    -> Adresse des VBL-Slot
    ;A_MOUSE_FLAG  -> Adresse von MOUSE_FLAG
    ;     MOUSE_FLAG = 0 ->Maus-Interuptbehandlung ein
    ;     MOUSE_FLAG <>0 ->Maus-Interuptbehandlung aus
    ;A_KBSHIFT     -> Adresse von Kbshift
    ;A_RUN         -> Adresse von Zeiger auf aktueller Basepage
A_KBDV:      dc.l 0
A_VBL_SLOT:     dc.l 0
A_MOUSE_FLAG:   dc.l 0
A_KBSHIFT:      dc.l 0
A_RUN:          dc.l 0
RESI_ENDE:
;************************
;Instalierung von CS_Maus
;************************
;gesammten Programmspeicher belegen
    EVEN
INSTALL:
        move.l   sp,a6        ;Adresse BASEPAGE
        lea      USTACK,sp    ;neuer Stack
        move.l   4(a6),a6     ;Speicher belegen
        move.l   $c(a6),a4
        adda.l   $14(a6),a4
        adda.l   $1c(a6),a4
        pea      256(a4)
        pea      (a6)
        clr.w -(sp)
        move.w   #74,-(sp)    ;Mshrink aufrufen
        trap #GEMDOS
        lea 12(sp),sp
;Adresse KBDVECS holen
        move #34,-(a7)        ;Kbdvbase aufrufen
        trap #XBIOS
        addq.l #2,a7
        move.l d0,A_KBDV      ;Adresse der Vektortabelle sichern
;CS_Maus schon installiert?
        move.l A_KBDV,a1      ;Adresse der Vektortabelle nach a1
```

→

```
        move.l 16(a1),a2      ;Adresse von Mausroutine in a2
        move.l a2,a0          ;a0 zum suchen
SP_1:  cmp.l #"RS11",-8(a0) ;CS_Maus schon vorhanden
        beq SP_2              ;ja, CS_MAUS init.
        cmp.l #"XBRA",-12(a0)   ;XBRA Kennung
        bne SP_4              ;CS_Maus installieren
                              ;Original Mausvektor oder kein XBRA Protokoll
        move.l -4(a0),a1      ;Adresse der nächsten Mausrutine
        move.l a1,a0          ;von a1 nach a0
        bra SP_1              ;weiter

;CS_Maus war schon installiert
*****************************
SP_2:  not.b -13(a0)          ;MAUS_OFF invertieren
        bne SP_7
;Meldung CS_Maus einschalten
        pea STRING_3
        move.w    #9,-(sp)
        trap #GEMDOS
        addq.l #6,sp
        bra SP_8              ;ende
;Meldung CS_Maus auschalten
SP_7:  pea STRING_4
        move.w    #9,-(sp)
        trap #GEMDOS
        addq.l #6,sp
;PRG beenden
SP_8:  clr.w   -(sp)
        trap #GEMDOS
;CS_MAUS instalieren
********************
;alten Vektor sichern
SP_4:  lea OLD_VEC,a0
        move.l a2,(a0)
;AES-anmelden
        move.w #10,d0         ;Applikation anmelden
        bsr AES_INIT          ;AES aufrufen
        tst.w AES_GLOBAL      ;starten aus AUTO-Ordner
        beq SP_6              ;ja, keine ap_version
;Vom Desktop starten
********************
;maus installieren
            lea DESKTOP,a0    ;Adresse von Desktop nach a0
        pea   (a0)
        move.w #38,-(sp)      ;Supexec aufrufen
        trap #XBIOS
        addq.l #6,sp
        bra SP_5
;Aus AUTO-Ordner starten
************************
;Unterprogramm im Supervisor-Modus ausführen
SP_6:  lea AUTO,a0   ;Adresse von auto nach a0
        pea   (a0)
        move.w #$26,-(sp) ;Supexec aufrufen
        trap #XBIOS
        addq.l #6,sp
;Adresse von MOUSE_FLAG holen
SP_5:  dc.w $a000   ;Adresse negative Line-A Variabelen
        sub.l #$153,a0  ;Adresse MOUSE-FLAG berechnen
        move.l a0,A_MOUSE_FLAG  ;MOUSE FLAG merken
;installierung i.o.
        pea STRING_1
        move.w    #9,-(sp)
        trap #GEMDOS
        addq.l #6,sp
;Speicherplatz für Mausroutine
;resident im Speicher halten
        clr.w -(sp)
        pea RESI_ENDE-RESI_ANFANG+256
        move.w #49,-(a7)      ;Ptermres aufrufen
        trap #GEMDOS
***************************
;Unterprogramme von CS_MAUS
***************************
;PRG im Supervisor-Modus Mausroutine installieren
*******************************************
;starten aus AUTO Ordner
;VBL installieren
AUTO:  move.w $454,d0         ;Anzahl der VBL Routinen (nvbls)
        lsl #2,d0             ;Anzahl*4
        move.l $456,a0        ;Zeiger auf VBL (_vblqueue)
        clr d1                ;Zähler
WEITER:  tst.l 4(a0,d1)       ;VBL Slot frei
        beq FREI              ;ja
        add #4,d1             ;nächster Slotpointer
        cmp.w d0,d1           ;alle slots abgefragt
        bne WEITER
;CS_Maus nicht initialisieren
        pea STRING_2          ;Alle Slots besetzt
        move.w    #9,-(sp)
        trap #GEMDOS
        addq.l #6,sp
        clr.w  -(sp)          ;PRG beenden
        trap #GEMDOS
FREI:  lea 4(a0,d1),a2        ;Adresse Slot in a2
        lea NEU_VBL,a1        ;Adresse neue Routine
        move.l a1,(a2)        ;neue Routine einhängen
        lea A_VBL_SLOT,a1     ;Slot Adresse
        move.l a2,(a1)        ;sichern
;Adresse von Kbshift ermitteln und
;Adresse von Zeiger auf aktueller Basepage
SP_10:    move.l #$e1b,A_KBSHIFT  ;Vorgabe für TOS 1.0 (Kbshift)
        move.l #$602c,A_RUN       ;Vorgabe für TOS 1.0 (Basepage)
        move.l $4f2,a0        ; (a0) _sysbase
        cmp.w #$0100,2(a0)    ;TOS 1.0
        beq SP_9              ;ja
        move.l 36(a0),a1      ; (a1) Adresse von Kbshift
        move.l a1,A_KBSHIFT   ;Adresse merken
        move.l 40(a0),a1      ; (a1) Adresse auf Zeiger der aktuellen Basepage
        move.l a1,A_RUN       ;Adresse merken
SP_9:  rts
```

→

```
;PRG im Supervisor-Modus Mausroutine installieren
*********************************************
;Starten vom Desktop
DESKTOP:
        move.l A_KBDV,a0      ;Keyboardadresse nach a0
        lea NEW_MOUSE,a1      ;Adresse neue Mausroutine
        move.l a1,16(a0)      ;neuen Vektor setzen

                              ;Kbshift und Zeiger auf
        bra SP_10             ;aktueller Basepage ermitteln

;AES aufrufen
*************
AES_INIT:
        lea CONTRL,a0         ;Adresse CONTRL nach a0
        move.w d0,(a0)+       ;Opcode
        clr.w (a0)+           ;einträge INIT_IN
        move.w #1,(a0)        ;einträge INIT_OUT
        clr.w (a0)+           ;einträge ADDR_IN
        clr.w (a0)            ;einträge ADDR_OUT

        move.l #AES_DATA,d1   ;Adresse AES-Array
        move.w #$c8,d0        ;AES-Aufruf
        trap #2
        rts


    DATA
    EVEN
STRING_1:
    dc.b 27,"E",27,"p"
    dc.b 13,10,"+****************************+"
    dc.b 13,10,"+                            +"
    dc.b 13,10,"+  Control-Shift-Maus  1.0   +"
    dc.b 13,10,"+          CS_MAUS           +"
    dc.b 13,10,"+                            +"
    dc.b 13,10,"+    7/91  Ralf Stachs       +"
    dc.b 13,10,"+        ST Computer         +"
    dc.b 13,10,"+                            +"
    dc.b 13,10,"+****************************+"
    dc.b 13,10
    dc.b 27,"q",0

    EVEN
STRING_2:
    dc.b 13,10,"Alle VBL-Slots besetzt",13, 10,0

    EVEN
STRING_3:
    dc.b 27,"E"
    dc.b 13,10,"*****************"
    dc.b 13,10,"*  CS_MAUS EIN  *"
    dc.b 13,10,"*****************",0

    EVEN
STRING_4:
    dc.b 27,"E",27,"p"
    dc.b 13,10,"*****************"
    dc.b 13,10,"*  CS_MAUS AUS  *"
    dc.b 13,10,"*****************"
    dc.b 27,"q",0

    EVEN
AES_DATA:
        dc.l CONTRL
        dc.l AES_GLOBAL
        dc.l INIT_IN
        dc.l INIT_OUT
        dc.l ADDR_IN
        dc.l ADDR_OUT

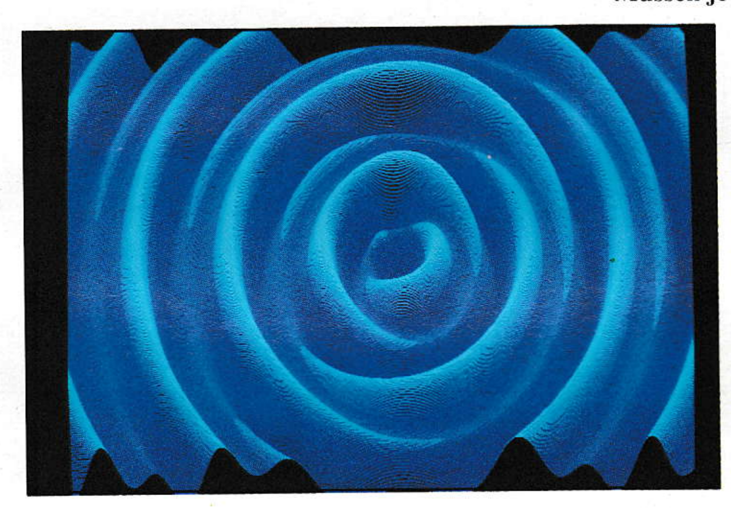
    BSS
AES_GLOBAL: ds.w 15
CONTRL:     ds.w 10
INIT_IN:    ds.w 128
INIT_OUT:   ds.w 128
ADDR_IN:    ds.l 128
ADDR_OUT:   ds.l 128

            ds.b 100
USTACK:     ds.w 0
```

Assembler ohne Grenzen

Arbeiten mit dem Coprozessor

Programme werden meistens dann in Assembler geschrieben, wenn sie möglichst schnell und kurz sein sollen. Müssen jedoch mathematische Berechnungen mit Fließkommazahlen durchgeführt werden, ist der Programmieraufwand gewaltig. In diesen Fällen sind höhere Programmiersprachen oft der einzige Ausweg. Durch den mathematischen Coprozessor des TT gehört dieses Problem nun der Vergangenheit an.



Das Prozessorgespann 68020/30 + 68881/2, wie z.B. im Atari TT, macht es möglich, auf Assembler-Ebene genauso selbstverständlich mit Fließkommazahlen umzugehen wie in Compilersprachen. Der Umfang an mathematischen Funktionen des Coprozessors entspricht dem von Hochsprachen und geht sogar teilweise darüber hinaus, wobei die Anwendung dieser Funktionen nicht wesentlich schwieriger ist. Wenn man also mathematische Anwendungen in Assembler programmieren möchte, braucht man nicht jedesmal das Rad neu zu erfinden, sondern kann, wie in anderen Programmiersprachen, auf einen leistungsfähigen Befehlssatz zurückgreifen.

Ohne Coprozessor ist die Programmierung einer Sinusfunktion eine recht komplizierte Angelegenheit, da der Sinus durch eine Potenzreihe angenähert werden muß. Wenn eine hohe Genauigkeit gewünscht wird, muß außerdem noch mit unhandlichen, 80 Bit großen Zahlen gearbeitet werden, wodurch das Ganze in eine wilde Bit-Schieberei ausartet. Eine entsprechende Routine würde mehrere Seiten Quelltext beanspruchen. Für Nichtmathematiker: Die Potenzreihe für eine Sinusberechnung mit ca. 3 genauen Nachkommastellen sieht folgendermaßen aus:

```
sin(x)=x-x^3/3!+x^5/5!-x^7/7!+x^9/9!
```

Mit Coprozessor sieht das gleiche so aus:

```
FSIN.X #x,FP0
```

Diese „Routine“ ist nicht nur ungleich kürzer, sondern auch um den Faktor 50 schneller als die andere, außerdem wird hier mit bis zu 19 Nachkommastellen gerechnet. Man erspart sich also viel Arbeit und hat gleichzeitig den Vorteil einer wesentlich höheren Geschwindigkeit. Es ist durchaus möglich, durch effiziente Programmierung des Coprozessors, mathematische Routinen um das 30-50fache zu beschleunigen. Selbst wenn man eine Hochsprache hat, die den Coprozessor unterstützt, lohnt es sich, diejenigen Programmteile, die Fließkommaarithmetik benutzen, in Assembler zu schreiben. Denn selbst gegenüber den schnellsten Compiler-Sprachen kann man noch leicht eine 50-200-prozentige Geschwindigkeitssteigerung erzielen. Außerdem sind Programmiersprachen, die den Coprozessor unterstützen, zur Zeit sehr teuer, und einige sind zudem recht fehlerhaft.

Wenn man über ein wenig Programmiererfahrung mit dem 68000 verfügt, ist es mit wenig Aufwand und Einarbeitungszeit möglich, auch komplizierte mathematische Probleme mit dem 68881/2 in Assembler zu lösen. Man braucht dafür nicht wesentlich länger als mit einer Hochsprache, aber durch die ungleich schnellere Ausführung der Programme wird man für diese Arbeit auf jeden Fall belohnt.

Ich möchte noch bemerken, daß ich hier nicht mehr auf die Programmierung der Prozessoren 60000-68030 eingehe. Es ist nicht unbedingt nötig, die zusätzlichen Befehle und Adressierungsarten des 68020/30 zu kennen, aber es erleichtert viele Dinge ungemein.

Extras!

Durch den 68881/2 wird der Befehlssatz des 68020/30 in drei Bereichen erweitert :

1. 49 Befehle (siehe Bild 1)
2. 11 Register :
   8 Datenregister FP0-FP7 (80 Bit)
   3 Kontrollregister FPCR, FPSR, FPIAR (32 Bit)
3. 4 Datentypen
   Single Precision Real (32 Bit)
   Double Precision Real (64 Bit)
   Extended Precision Real (96 Bit)
   Packed Decimal String Real (96 Bit)

Wichtig ist dabei, daß die physikalische Trennung der beiden Prozessoren für den Programmierer nicht transparent ist, d.h. die Entscheidung, welcher Prozessor einen Befehl ausführen muß, wird von der Hardware getroffen. Dies gilt nicht für den 68000-Prozessor, da er keine Coprozes-

Einstellige Operationen	Beispiele
FCOS,FACOS,FCOSH,FSIN,FASIN,FSINH Bedeutung ist FTAN,FATAN,FTANH,FATANH direkt abzulesen FLOG10,FLOG2,FLOGN ; FABS,FNEG,FSQRT	FATANH.X FP0,FP1 FP1:=Arcustangens Hyperbolicus von FP0
FETOX e^x FETOXM1 (e^x)-1 FLOGNP1 LN(x+1) FTENTOX 10^x FTWOTOX 2^x	FP0=3 FTENTOX.X FP0 FP0=10^3=1000
FGETEXP Extrahiert den Exponenten der binären Floatingpointdarstellung des Operanden FGETMAN Extrahiert die Mantisse der binären Floatingpointdarstellung des Operanden	FP0=256 FGETEXP.X FP0 FP0=8 da 256=2^8 FP0=256 FGETMAN.X FP0 FP0=1 da 256=1*2^8
FINT Berechnet Integer unter Beachtung des Rundungsmodus FINTRZ Nachkommaanteil wird abgeschnitten	FP0=2.51 (nächste Zahl) FINT.X FP0 FP0=3.0 FINTRZ.X FP0 FP0=2.0
FSINCOS Berechnet Sinus u. Cosinus gleichzeitig	FSINCOS.X FPm,FPc:FPs

Zweistellige Operationen	Beispiele
FADD,FSUB,FMUL,FDIV +,-,*,/	FP0=2,FP1=5 FDIV.X FP0,FP1 FP1=2.5
FSGLMUL,FSGLDIV *,/ Das Ergebnis wird nach Single Precision gerundet unabhängig von der Einstellung	
FMOD Modulo Funktion Ziel-(Quelle*(INT(Ziel/Quelle)) INT schneidet Nachkommastellen ab FREM IEEE Remainder INT rundet hier zur nächsten Zahl Das Quotient Byte des FPSR enthält nach Ausführung dieser Funktionen das Ergebnis von INT(Ziel/Quelle)	FP0=5,FP1=3 FMOD.X FP1,FP0 FP0=2 Quotient Byte=1 FP0=5,FP1=3 FREM.X FP1,FP0 FP0=-1 Quotient Byte=2
FSCALE INT(Quelle) wird zum Exponenten des Zieloperanden addiert INT schneidet Nachkommastellen ab	FP0=2.5,FP1=2 FSCALE.X FP1,FP0 FP0=2.5*2^(0+2)=10
FCMP Entspricht in der Funktionsweise dem CMP des 68000	FP0=5,FP1=4 FCMP.X FP1,FP0 N Bit = 1

Sonstige Befehle	Beispiele
FBcc Funktionsweise wie Bcc FDBcc " DBcc FScc " Scc FTRAPcc " TRAPcc Hier werden jedoch die FPU-Flags abgefragt	FBOGT.L <LABEL> FDBNGL.W D0,<LABEL> FSUN D0 FTRAPGE
FTST Funktionsweise wie TST	FTST.X FP0
FSAVE Sichert Koprozessorstatus FRESTORE Restauriert Koprozessorstatus Diese Befehle benötigt man für die Prozessumschaltung bei Multitasking	FSAVE (A0) FRESTORE (A0)
FMOVE.L Tranportiert Daten von und zu den Floating Point Controlregistern FMOVEM.L "	FMOVE.L #$0f000000,FPSR FMOVEM.L FPSR/FPCR,-(sp)
FMOVE.f Transportiert Daten von und zu den Floating Point Datenregistern FMOVEM.X "	FMOVE.W D1,FP7 FMOVEM.X (sp)+,FP0-FP4
FMOVECR.X Transportiert Konstante aus internem ROM in ein Floating Point Datenregister	FMOVECR.X #0,FP7 FP7=PI=3.1415926...

Bild 1: Der Befehlssatz des 68881/2

sorschnittstelle besitzt. Bei einem Mega ST/E wird der Coprozessor, falls vorhanden, über Hardware-Adressen angesprochen. Dadurch wird die Programmierung sehr kompliziert und damit unattraktiv. Bei einem Rechner mit 68020/30 und 68881/2 hat man also sozusagen, „einen" Prozessor mit erweitertem Befehlssatz und zusätzlichen Registern. Der Coprozessor kann insgesamt mit 7 Datentypen arbeiten, welche allerdings nur für die Kommunikation mit der Außenwelt bestimmt sind, da intern immer mit höchster Genauigkeit und einem eigenen Format gearbeitet wird. Das heißt, es müssen alle Operanden in das interne Format gewandelt werden, bevor mit ihnen gerechnet werden kann. Darüber braucht man sich jedoch keine Gedanken zu machen, da der Coprozessor sämtliche Formatumwandlungen vollkommen automatisch erledigt. Dadurch ist es möglich, Berechnungen mit beliebigen Zahlenformaten durchzuführen, ohne daß man sich um deren Anpassung kümmern muß. Notwendig ist nur die Angabe des Datentypes des Quelloperanden. Der Zieloperand ist bei arithmetischen Operationen immer ein FPU-Datenregister, wie aus Bild 1 zu ersehen ist.

Der Aufbau der 4 neuen Datentypen ist in Bild 2 veranschaulicht. Bei SINGLE, DOUBLE und EXTENDED ist der innere Aufbau jedoch für die Programmierung nicht von Bedeutung, hier aber der Vollständigkeit halber angegeben. Wichtig ist nur, welche Genauigkeit die Datentypen haben und wieviel Platz sie im Speicher benötigen. Zum Rechnen können alle Datentypen von Byte bis Extended benutzt werden, nur das Packed-Decimal-String Real Format sollte nicht verwendet werden. Es ist, wie der Name schon sagt, einem String sehr ähnlich und daher leicht in einen solchen zu verwandeln, wie das kleine Programm zeigt. Die umgekehrte Richtung geht so ähnlich, allerdings muß man dann eventuell vorhandene Fehler abfangen. Der Befehl *FMOVE.P FPm, <ea>{#k}* bzw. *{Dn}* bietet die Möglichkeit zu bestimmen, mit wievielen Stellen eine Zahl ausgegeben werden soll. Man hat dabei die Wahl zwischen der Anzahl der Stellen rechts vom Komma, wobei der Exponent berücksichtigt wird, und der Anzahl der Stellen in der Mantisse. Mit Hilfe dieser Informationen steht der Programmierung beliebiger mathematischer Funktionen nichts mehr im Wege (siehe Bilder 1-3).

Beispiele:

```
D0.B=SQRT(D0.B)
FSQRT.B    D0,FP0
FMOVE.B    FP0,D0
D1.W=SIN(Pi/D0.B)
FMOVECR.X #0,FP0
FDIV.B     D0,FP0
FSIN.X     FP0
FMOVE.W    FP0,D1
```

Es soll folgender Graph gezeichnet werden:

```
20*e^(SIN(x/k1)+COS(x/k2))+k3
```

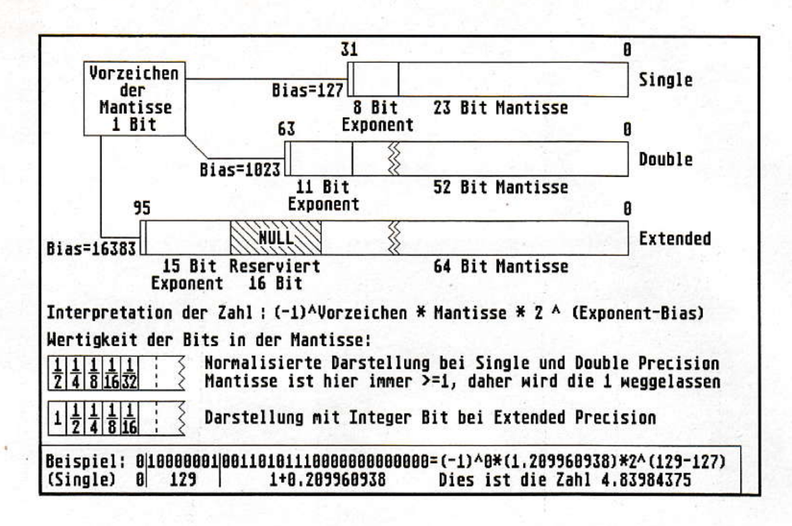
Die Konstanten k1, k2, k3 (16 Bit) stehen ab der Stelle, auf die A0 zeigt. Die Variable x läuft von 0 bis 640.

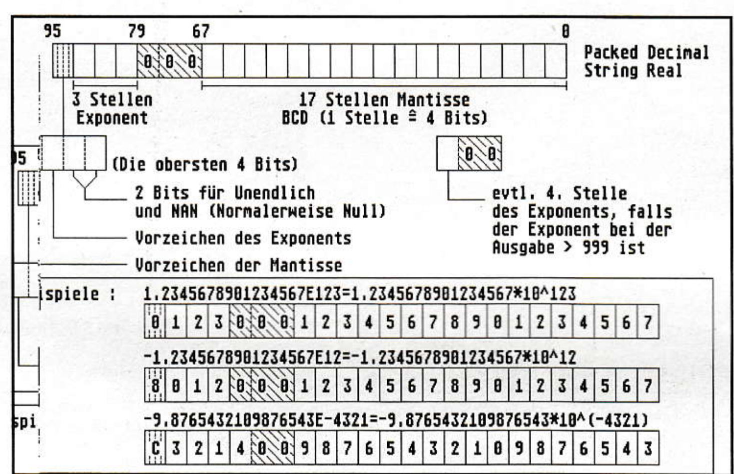
```
MOVEQ       #0,D0
  FMOVE.B    #20,FP0
LOOP:
  FMOVE.W     D0,FP1
  FMOVE.W    D0,FP2
  FDIV.W     (A0),FP1
  FDIV.W     2(A0),FP2
  FSIN.X     FP1
  FCOS.X    FP2
  FADD.X     FP2,FP1
  FETOX.X    FP1
  FMUL.X     FP0,FP1
  FADD.W     4(A0),FP1
  FMOVE.W    FP1,D1
BSR         PUNKT
ADDQ.W      #1,D0
CMPI.W      #640,D0
BNE         LOOP
```

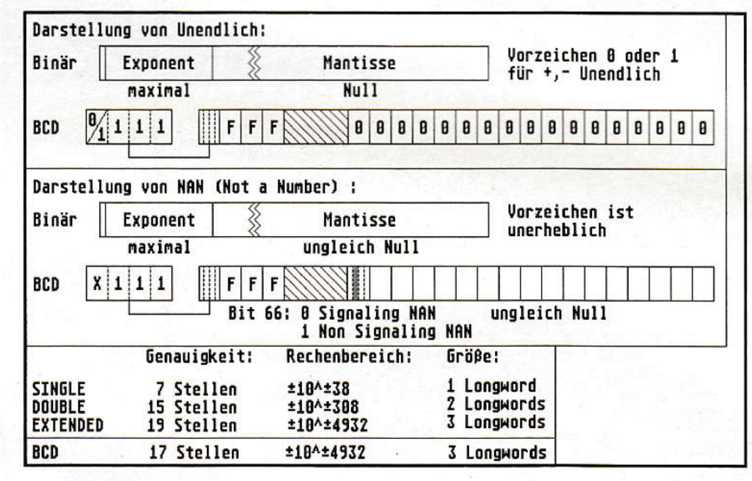
(siehe auch das Rosetten- und das Wellen-Programm auf der Leserservicediskette)

Bedingte Verzweigungen

Genau wie bei der CPU, gibt es auch bei der FPU Befehle, deren Ausführung von Bedingungen abhängt. Dies sind die Befehle *FBcc,FDBcc,FScc* und *FTRAPcc.*







	Genauigkeit:	Rechenbereich:	Größe:
SINGLE	7 Stellen	±10^±38	1 Longword
DOUBLE	15 Stellen	±10^±308	2 Longwords
EXTENDED	19 Stellen	±10^±4932	3 Longwords
BCD	17 Stellen	±10^±4932	3 Longwords

Bild 2: Aufbau der neuen Datentypen

Der Unterschied zu den entsprechenden Befehlen der CPU besteht darin, daß hier das Condition-Code-Byte des FPSR ausgewertet wird. Da die Bedeutung der Bits N, Z und I klar sein dürfte, werde ich nur das NAN-Bit erläutern: Das Ergebnis einer arithmetischen Operation (z.B. *FADD*) kann in drei Fälle aufgespalten werden. Der 1. Fall (Normalfall) tritt ein, wenn alle Operanden im Definitionsbereich der jeweiligen Funktion liegen. Im 2. Fall kann die Funktion zwar nicht errechnet werden, das Ergebnis ist jedoch mathematisch festgelegt, z.B. 0+Unendlich = Unendlich, -1*Unendlich = -Unendlich, LN(0) = Unendlich. Der letzte Fall tritt dann auf, wenn das Ergebnis mathematisch nicht zu bestimmen ist, z.B. -Unendlich+Unendlich, 0/0, LN(-1). Wenn dies passiert, wird das NAN-Bit gesetzt, und gleichzeitig wird dem Zieloperanden ein sogenanntes NAN-Format zugewiesen. Sollte bei einer Operation einer der Operanden gleich NAN sein, so ist das Ergebnis ebenfalls NAN, und das NAN-Bit wird in diesem Fall auch gesetzt. Das I-Bit wird übrigens von den Bedingungscodes nicht berücksichtigt. Wenn es für eigene Zwecke benötigt wird, kann man es durch ein „*FMOVE.L FPSR, Dn*“ in ein CPU-Datenregister schieben und dann selbst auswerten. Es gibt zwei Gruppen von Bedingungscodes (Bild 4), die sich voneinander darin unterscheiden, daß die einen das BSUN-Exception-Bit setzen können und die anderen nicht (BSUN s.u.). Die sogenannten *IEEE Non-Aware-Bedingungscodes* sind jedoch nur für denjenigen von Interesse, der tiefer in diese Materie einsteigen will. Nur noch eine Bemerkung hierzu : Durch die Existenz von NANs, also Zahlen, die keine sind, werden die Bedingungscodes etwas komplizierter als nötig. Das Gegenteil von „Less Than“ ist dann nicht mehr „Greater Than or Equal“, sondern „Not Less Than“.

Nun noch ein abschließendes Beispiel hierzu: Die folgenden Zeilen realisieren eine Schleife, die von 0 bis 2*Pi in Schritten der Größe 0.1 zählt.

```
  FMOVECR.X  #0,FP0
  FMUL.B     #2,FP0
  FMOVE.B    #1,FP1
  FDIV.B     #10,FP1
  FMOVE.B    #0,FP2
LOOP:

  (beliebiges Programm)

  FADD.X     FP1,FP2
  FCMP.X     FP0,FP2
  FBOLE      LOOP
```

Fehler aufgetreten?

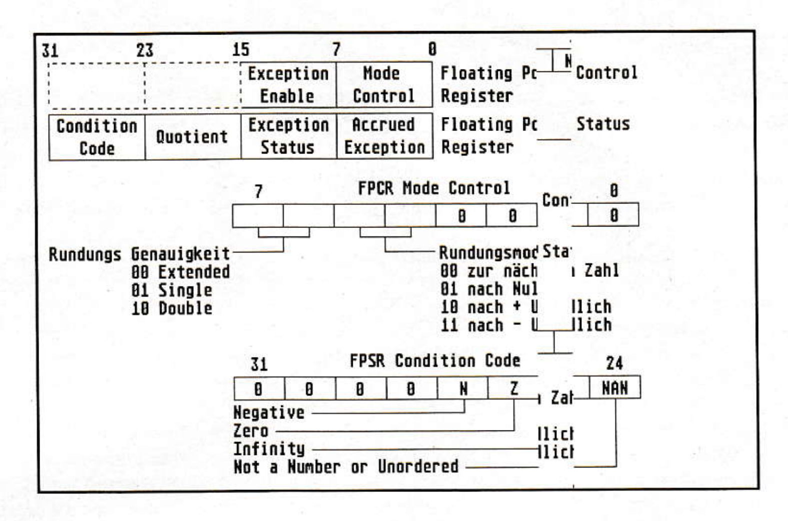
Um festzustellen, ob während einer Rechnung ein Fehler aufgetreten ist, benötigt man das Status- und das Control-Register der FPU (Bild 5), genauer gesagt das *Exception-Enable-* und das *Exception-Status-Byte*. Beide Bytes haben den gleichen Aufbau, jedoch eine unterschiedliche Bedeutung. Wenn irgendein Fehler auftritt, wird im Exception Status Byte das entsprechende Bit gesetzt. Das Exception Enable-Byte hingegen stellt eine Maske dar, die angibt, welche Fehler eine Exception-Behandlung auslösen können. Ist z.B. im Exception-Enable-Byte das DZ-Bit gesetzt, wird im Falle einer Division durch Null eine Ausnahmebehandlung (Exception) über Vektor 50 ausgelöst. Dadurch hat man die Möglichkeit, für jeden Fehler eine Routine im Betriebssystem zu installieren, die eine entsprechende Fehlerbehandlung durchführt. Man könnte zum Beispiel eine Dialogbox ausgeben, die es einem ermöglicht, das Programm abzubrechen oder es mit korrigierten Parametern fortzusetzen. Da das TOS diese Vektoren nicht belegt, sollten normalerweise alle Bits im Exception-Enable-Byte auf Null gesetzt sein.

Das Exception-Status-Byte stellt jedoch nur eine Momentaufnahme dar, d.h. beim nächsten Befehl werden unter Umständen alle Bits verändert. Dadurch wäre man gezwungen, nach jedem Befehl abzufra-

IEEE Non-Aware Floating Point Conditional Operations können BSUN exception Bit im FPSR setzen		
SF	Signaling False	0
SEQ	Signaling Equal	Z
GT	Greater Than	~(NAN|Z|N)
GE	Greater Than or Equal	Z|~(NAN|N)
LT	Less Than	N&~(NAN|Z)
LE	Less Than or Equal	Z|(N&~NAN)
GL	Greater or Less Than	~(NAN|Z)
GLE	Greater, Less, or Equal	~NAN
NGLE	Not(Greater, Less, or Equal)	NAN
NGL	Not(Greater or Less Than)	NAN|Z
NLE	Not(Less Than or Equal)	NAN|~(N|Z)
NLT	Not Less Than	NAN|(Z|~N)
NGE	Not(Greater Than or Equal)	NAN|(N&~Z)
NGT	Not Greater Than	NAN|Z|N
SNE	Signaling Not Equal	~Z
ST	Signaling True	1
		(~ NOT ; | OR ; & AND)

IEEE Aware Floating Point Conditional Operations		
F	False	0
EQ	Equal	Z
OGT	Ordered Greater Than	~(NAN|Z|N)
OGE	Ordered Greater Than or Equal	Z|~(NAN|N)
OLT	Ordered Less Than	N&~(NAN|Z)
OLE	Unordered or Less Than or Equal	Z|(N&~NAN)
OGL	Ordered Greater or Less Than	~(NAN|Z)
OR	Ordered	~NAN
UN	Unordered	NAN
UEQ	Unordered or Equal	NAN|Z
UGT	Unordered or Greater Than	NAN|~(N|Z)
UGE	Unordered or Greater Than or Equal	NAN|(Z|~N)
ULT	Unordered or Less Than	NAN|(N&~Z)
ULE	Unordered or Less Than or Equal	NAN|Z|N
NE	Not Equal	~Z
T	True	1

Bild 3+4: Es gibt zwei Gruppen von Bedingungscodes



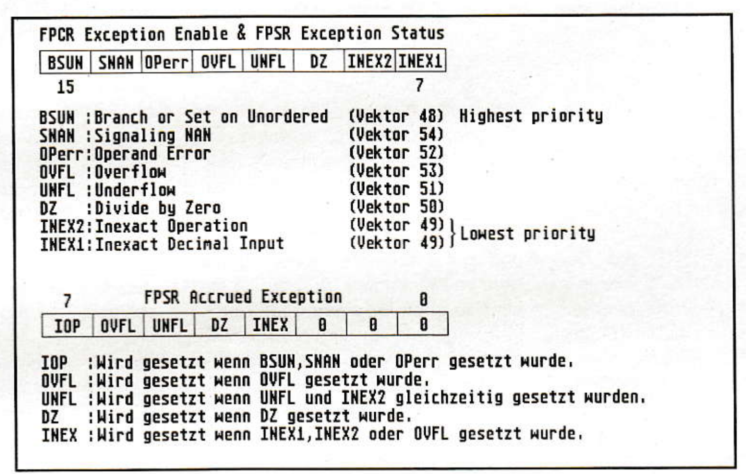


Bild 5: Das Status -und Control-Register der FPU

gen, ob irgendein Fehler aufgetreten ist. Dies ist aber in den meisten Fällen nicht sinnvoll und auch zu umständlich. Das *Accrued-Exception-Byte* schafft hier Abhilfe, da es sich das Auftreten eines Fehlers beliebig lange merkt. Jedes einmal gesetzte Bit kann nur vom Benutzer durch einen *FMOVE*-Befehl gelöscht werden. Wenn man also dieses Byte vor einer längeren Berechnung löscht, kann hinterher festgestellt werden, ob ein bestimmter Fehler aufgetreten ist, um dann zu entscheiden, ob das Ergebnis gültig ist oder nicht. Es ist dann allerdings nicht mehr möglich zu rekonstruieren, welcher Befehl den Fehler ausgelöst hat.

Die Bedeutung der verschiedenen Bits im einzelnen:

BSUN: Dieses Bit wird gesetzt, wenn das NAN-Bit gesetzt ist und einer der Befehle FBcc, FDBcc, FScc und FTRAPcc im Zusammenhang mit einer IEEE Non-Aware Bedingung benutzt wird.
SNAN: Durch das sogenannte „Signaling Nan"-Format ist es möglich, eigene Datenformate zu kreieren.
OPerr: Dieser Fehler tritt auf, wenn bei bestimmten arithmetischen Operationen der Quelloperand außerhalb des mathematischen Definitionsbereiches liegt. OPerr wird ebenfalls gesetzt, wenn das NAN-Bit gesetzt wird. Beispiele: ACOS(x) ist nicht definiert für x=+-Unendlich,x>1 und x<-1; SQRT(x) ist nicht definiert für x<0 und x=-Unendlich; OPerr wird auch gesetzt, wenn bei FMOVE.B/W/L der Quelloperand nicht in das Integerformat paßt oder wenn er NAN ist.
OVFL: Ein Overflow tritt dann auf, wenn beim Arbeiten mit single-, double- oder extended-precision-Zahlen, unter Beachtung der Rundungsgenauigkeit, die zuzuweisende Zahl zu groß ist, um im Zieloperanden gespeichert werden zu können.
UNFL: Ähnlich wie OVFL, aber hier ist die Zahl zu klein, um noch dargestellt werden zu können. Bei B/W/L wird die Null zugewiesen ohne UNFL zu setzen.
DZ: Division durch Null. Tritt aber auch in folgenden Fällen ein: FATANH(+-1), FLOGxx(0), FLOGNP1(-1).
INEX1: tritt dann auf, wenn eine dezimale Zahl (.P Format) eingelesen wird, die binär nicht genau dargestellt werden kann, z.B:0.1.
INEX2: Sobald eine Zahl gerundet wird, also Nachkommastellen abgeschnitten werden, wird INEX2 gesetzt. Dies hängt natürlich von der Rundungsgenauigkeit ab.

Geht es noch schneller?

Diese Frage kann positiv beantwortet werden, denn trotz der enormen Geschwindigkeit der FPU ist es möglich, Programme mit einfachen Mitteln wirkungsvoll zu beschleunigen. Ein bekanntes Mittel zur Beschleunigung von Programmen ist die registeroptimierte Programmierung, welche natürlich auch bei der FPU nicht wirkungslos ist, denn es fällt nicht nur ein Lesezyklus weg, sondern auch die Umwandlung der Operanden in das interne 80-Bit-Format.

Die zweite Möglichkeit zur Optimierung liegt in der Tatsache begründet, daß die FPU ein eigenständiger Prozessor ist, denn während die FPU eine arithmetische Berechnung durchführt, kann die CPU ganz normal weiterarbeiten. Dies ist jedoch nur dann machbar, wenn kein Prozessor auf das Ergebnis einer Berechnung des anderen wartet und kein Register von beiden benutzt wird. Die parallele Ausführung von Befehlen scheitert auch dann, wenn ein zweiter FPU-Befehl gestartet wird, während der erste noch bearbeitet wird. In diesem Fall warten beide Prozessoren auf die Fertigstellung des ersten Befehls. Zur Parallelverarbeitung eignen sich am besten diejenigen FPU-Befehle, welche die meiste Ausführungszeit benötigen. So kann die CPU beispielsweise 10 Multiplikationen durchführen, während die FPU einen Sinus berechnet. Dadurch wird die Sinusberechnung praktisch in Nullzeit erledigt. Diese effektive Art der Optimierung

```
Einstellige Operationen :        Zweistellige Operationen :
  FXXX.f   <ea>,FPn               FXXX.f   <ea>,FPn
  FXXX.X   FPm,FPn                FXXX.X   FPm,FPn
  FXXX.X   FPn

FMOVE und FMOVEM für Floating Point Data Register :
  FMOVE.f  <ea>,FPn               FMOVEM.X <list>,<ea>
  FMOVE.f  FPm,<ea>               FMOVEM.X Dn,<ea>
  FMOVE.X  FPm,FPn                FMOVEM.X <ea>,<list>
  FMOVE.P  FPm,<ea>{Dn}           FMOVEM.X <ea>,Dn
  FMOVE.P  FPm,<ea>{#k}
 <list> : Liste von Floating Point Daten Registern
     z.B: FP0-FP7 oder FP0-FP3/FP5-FP7 oder FP0/FP2/FP4
  Unteres Byte von Dn         |7|6|5|4|3|2|1|0| Predecrement
  wird bei FMOVEM         FPn<
  als Maske benutzt.          |0|1|2|3|4|5|6|7| Sonst
                                unteres Byte

FMOVE und FMOVEM für Floating Point Control Register :
  FMOVE.L  <ea>,FPcr              FMOVEM.L <list>,<ea>
  FMOVE.L  FPcr,<ea>              FMOVEM.L <ea>,<list>
 <list> : Liste von Floating Point Control Registern
     z.B: FPCR/FPSR/FPIAR oder FPCR/FPSR
```

```
FMOVECR.X #cc,FPn

Transportiert
Konstante in FPU
Register

$00     PI
$0B     LOG10(2)
$0C     e
$0D     LOG2(e)
$0E     LOG10(e)
$0F     0.0
$30     LN(2)
$31     LN(10)
$32     10^0
$33     10^1
$34     10^2
$35     10^4
$36     10^8
 :        :
$3D     10^1024
$3E     10^2048
$3F     10^4096
```

```
f steht für :

B  Byte Integer
W  Word Integer
L  Longword Integer
S  Single Precision Real
D  Double Precision Real
X  Extended Precision Real
P  Packed Decimal String Real
```

```
{Dn} oder {#k} =
  0 bis -64  Anzahl der signifikanten
(≙0 bis  64) Dezimalstellen rechts
             vom Komma (Fortran F Format)
  1 bis 17   Anzahl der signifikanten
             Dezimalstellen in der
             Mantisse  (Fortran E Format)
 18 bis 63   Löst Operand Error (OPerr) aus
             ansonsten wie 17

Die Zahlen werden hierbei gerundet !
```

Adressierungsarten für <ea> :

Syntax	Erlaubt	Syntax	Erlaubt
Dn	◆	xxx.W	◆
An	*	xxx.L	◆
(An)	◆	#<data>	◆ □
(An)+	◆		
-(An)	◆		
(d16,An)	◆	(d16,PC)	◆ □
(d8,An,Xn)	◆	(d8,PC,Xn)	◆ □
(bd,An,Xn)	◆	(bd,PC,Xn)	◆ □
([bd,An,Xn],od)	◆	([bd,PC,Xn],od)	◆ □
([bd,An],Xn,od)	◆	([bd,PC],Xn,od)	◆ □

* Nur erlaubt bei FMOVE und FMOVEM für Floating Point Control Register

□ Nicht erlaubt als Zieloperand

Bild 6: Der FMOVE-Befehl beim 68882

ist aber leider nur selten einsetzbar, da die genannten Einschränkungen bei der Programmierung oft auftreten.

Darüber hinaus ist der 68882, im Gegensatz zum 68881, in der Lage, auch eigene Befehle parallel zu verarbeiten oder deren Ausführung teilweise zu überlappen. Dies sind vor allem arithmetische Operationen im Zusammenhang mit FMOVE-Befehlen, welche keine Formatumwandlungen vornehmen müssen, wie z.B.: *FMOVE FPm, FPn* oder *FMOVE.X <ea>,FPm.* Man sollte immer versuchen, schnelle *FMOVE*-Befehle mit schnellen arithmetischen Befehlen zu kombinieren und umgekehrt. Die parallele Verarbeitung ist nicht möglich, wenn Registerkonflikte auftreten und wenn mit den Datentypen B, W, L, P gearbeitet wird. Folgende Befehle können nicht oder nur ganz minimal parallel ausgeführt werden : *FBcc, FDBcc, FScc, FTRAPcc, FMOVEM.*
Beispiel für eine Optimierung auf dem 68882:

Programmschleifen sollten, wenn möglich, teilweise 'aufgerollt' werden.

```
FMOVE.X (A0)+,FP0
FMUL.X  FP2,FP0
FMOVE.X FP0, (A1)+
```

Die folgende Routine ist ca. 12 % schneller:

```
FMOVE.X (A0)+,FP0
FMOVE.X (A0)+,FP1
FMUL.X  FP2,FP0
FMUL.X  FP2,FP1
FMOVE.X FP0, (A1)+
FMOVE.X FP1, (A1)+
```

Diese Optimierungen bringen natürlich nichts, wenn die programmierten Funktionen nicht bereits auf mathematischem Wege vereinfacht wurden. Man sollte möglichst versuchen, die Anzahl der trigonometrischen Operationen so klein wie möglich zu halten. Zur Vereinfachung von Polynomen eignet sich das sogenannte Horner-Schema ganz besonders.

Beispiel :

2*x^3-4*x^2+x-1 (5 Multiplikationen)
-> -1+(1+(-4+2*x)*x)*x (3 Multiplikationen).

Die Demoprogramme

Das hier abgedruckte Rosette- und das Wellen-Programm auf der Leserservicediskette (siehe auch Titelbild) sollen verdeutlichen, wie einfach die Programmierung des Coprozessors ist. Da ich hier auf die Funktionsweise der Programme nicht eingehen möchte, habe ich die beiden GFA-BASIC-Listings beigefügt, welche die grundsätzliche Arbeitsweise verdeutlichen. Wie man sieht, sind die eigentlichen Hauptprogramme sehr kurz im Gegensatz zu den Eingaberoutinen, welche man allerdings universell einsetzen kann. Wer nicht viel tippen möchte, kann den Eingabeteil inklusive der entsprechenden Unterprogramme weglassen, muß dann allerdings die Parameter direkt eintragen und jedesmal neu assemblieren. Es kann jeder Assembler verwendet werden, der die Befehle des 68030 und des 68882 unterstützt, wie z.B. der MAS 68K. Allerdings sollte man den dazugehörigen Linker TLINK.TTP nicht verwenden, da er ab bestimmten Quellcodegrößen Fehler macht.

Die beiden Demos zeigen, wie groß die Beschleunigung von mathematischen Programmen durch den Coprozessor ist, und daß sich die Beschäftigung mit dessen Programmierung lohnt. Das Wellen-Programm benötigt beispielsweise für eine Animation mit 80 Bildern ca. 10 Minuten. Ein entsprechendes Programm in TURBO C benötigt ohne Coprozessor 3 Stunden, auf einem normalen ST sogar 11 Stunden. Auf einem Atari TT ist es also möglich, selbst aufwendige Animationen in einem Bruchteil der bisher üblichen Zeit zu berechnen.

Zum Schluß!

Diese kleine Einführung in die Programmierung des 68881/2 kann und will kein Ersatz für ein ausführliches Handbuch sein, aber es sollte trotzdem möglich sein, mit Hilfe dieser Informationen eigene Programme zu erstellen. Die Programmierung des Coprozessors ist wie gesagt sehr einfach, da er einem alle Aufgaben, die kompliziert sind oder viel Arbeit machen, einfach abnimmt; es ist jedoch ein umfangreiches Detailwissen vonnöten, um alle Möglichkeiten des 68881/2 nutzen zu können. Ich habe hier versucht, die wichtigsten Informationen zusammenzustellen, bin mir aber bewußt, daß diese Auswahl nicht jeden zufriedenstellen wird. Daher empfehle ich jedem, der sich intensiver mit dieser Materie beschäftigen möchte, entsprechende Fachliteratur zu erwerben. Für Fragen und Anregungen hier noch meine Adresse:

Jochen Fischer
Gierlichstr. 2
W-5120 Herzogenrath

Literatur:
Steve Williams
"68030 Assembly Language Reference"
Addison-Wesley Publishing Company Inc.

```
* Umwandlung des Packed Decimal String Real
* Formates in einen String

  lea       zahl,a0        ;Puffer für Zahl
; fmove.f   <ea>,fpn
; fmove.p   fpn,(a0){#k}
  lea       string,a1      ;Puffer für String
  btst.b    #7,(a0)        ;Zahl negativ ?
  beq       positiv        ;nein
  move.b    #'-',(a1)+     ;ja
positiv:
  move.b    3(a0),d0       ;erste Ziffer
  unpk      d0,d1,#$3030 ;in ASCII wandeln
  move.b    d1,(a1)+       ;in String schreiben
  move.b    #'.',(a1)+     ;Komma setzen
  moveq     #4,d2          ;Offset
  moveq     #7,d3          ;Zähler
m_loop:
  move.b    (a0,d2.w),d0 ;2 Ziffern holen
  unpk      d0,d1,#$3030 ;in ASCII wandeln
  move.w    d1,(a1)+       ;in String schreiben
  addq.w    #1,d2          ;Offset erhöhen
  dbf       d3,m_loop      ;bis zur letzten Ziffer
  move.b    #'e',(a1)+
  btst.b    #6,(a0)        ;Exponent negativ?
  beq       positiv2       ;nein
  move.b    #'-',(a1)+     ;ja
positiv2:
  bfextu    2(a0){0,4},d0;1. Stelle d. Exponents
  lsl.b     #4,d0
  bfextu    (a0){4,4},d1 ;2. Stelle d. Exponents
  or.b      d1,d0          ;1. u. 2. Stelle in D0
  unpk      d0,d1,#$3030 ;in ASCII wandeln
  move.w    d1,(a1)+
  move.b    1(a0),d0
  unpk      d0,d1,#$3030 ;3. und 4. Stelle
  move.w    d1,(a1)+
* ----------------------------------------------
zahl:       dc.l        0,0,0
*
            bss
string:     ds.b        20
```

```
******************************
*                            *
*  ROSETTE.TOS               *
*                            *
*  by Jochen Fischer         *
*                            *
*  (c) 1991 MAXON Computer   *
******************************

macro Cconws str
  movem.l   d1-d2/a0-a2,-(sp) ;Ausgabe des
  pea       str               ;Strings str
  move.w    #9,-(sp)          ;unter Beachtung
  trap      #1                ;der ESC-Sequenzen
  addq.l    #6,sp
  movem.l   (sp)+,d1-d2/a0-a2
endm
macro Cconrs buf
  movem.l   d1-d2/a0-a2,-(sp) ;Lesen des
  pea       buf               ;Strings str
  move.w    #10,-(sp)         ;unter Beachtung
  trap      #1                ;der ESC-Sequenzen
  addq.l    #6,sp
  movem.l   (sp)+,d1-d2/a0-a2
endm
macro Cconin
  movem.l   d1-d2/a0-a2,-(sp) ;Warten auf
  move.w    #1,-(sp)          ;einen
  trap      #1                ;Tastendruck
  addq.l    #2,sp             ;d0=Zeichen
  movem.l   (sp)+,d1-d2/a0-a2
endm
macro rd_int buf,frage,len,pos ;Read Integer
  pea       buf(pc)           ;Puffer für Eingabe
  pea       frage(pc)         ;Zeiger auf Frage
  move.w    len,-(sp)         ;Max. Eingabelänge
  move.w    pos,-(sp)         ;(X,Y)-Position
  bsr       lies_int
  adda.l    #12,sp
endm
macro rd_card buf,frage,len,pos ;Read Cardinal
  pea       buf(pc)           ;s.o.
  pea       frage(pc)
  move.w    len,-(sp)
  move.w    pos,-(sp)
  bsr       lies_card
  adda.l    #12,sp
endm
* ----------------------------------------------
init        equ $0
maus_off    equ $a
maus_on     equ $9
a_line      equ $3
* ----------------------------------------------
*                  Initialisierung
* ----------------------------------------------
  text
  movem.l   d0-d7/a0-a6,-(sp)       ;CPU Reg. und
  fmovem.l  fpcr/fpsr/fpiar,-(sp)   ;alle FPU Reg.
  fmovem.x  fp0-fp7,-(sp)           ;sichern
  aline     init                    ;line_a init
  lea       a_zeiger,a1             ;Line_a Zeiger
  move.l    a0,(a1)                 ;sichern
  move.w    -$c(a0),d0              ;x_max
  lsr.w     #1,d0                   ;d0=x_halbe
  move.w    -$4(a0),d1              ;y_max
  lsr.w     #1,d1                   ;d1=y_halbe
  lea       x_hlb(pc),a0            ;x und y_halbe
  move.w    d0,(a0)                 ;sichern
  move.w    d1,2(a0)
  Cconws    clr_scr(pc)
* ----------------------------------------------
*           Eingabeteil
* ----------------------------------------------
  rd_card   Antwort,Frage1,#4,#$0202 ;Rad.1 lesen
  cmp.w     d1,d0           ;Radius 1<=y_halbe ?
  bge       e1
  move.w    d0,4(a0)        ;ja -> nehme Eingabe
  bra       w1
e1:
  move.w    2(a0),4(a0)     ;nein->nehme y_halbe/2
  lsr.w     4(a0)
w1:
  move.w    2(a0),d2        ;y_halbe
  sub.w     4(a0),d2        ;y_halbe-Radius 1
  rd_card   Antwort,Frage2,#4,#$0402 ;Rad.2 lesen
  cmp.w     d2,d0           ;Rad.2<=y_halbe-Rad.1 ?
  bge       e2
  move.w    d0,6(a0)        ;ja -> nehme Eingabe
  bra       w2
e2:
  move.w    d2,6(a0)    ;nein->nehme y_halbe-Rad.1
w2:
  rd_int    Antwort,Frage3,#5,#$0602 ;lese Freq.1
  move.w    d0,8(a0)
  rd_int    Antwort,Frage4,#5,#$0802 ;lese Freq.2
  move.w    d0,10(a0)
  rd_card   Antwort,Frage5,#3,#$0a02 ;lese Konst1
  move.w    d0,12(a0)
  rd_card   Antwort,Frage6,#3,#$0c02 ;lese Konst2
  move.w    d0,14(a0)
  rd_card   Antwort,Frage7,#5,#$0e02 ;Genauigkeit
  move.w    d0,16(a0)
  Cconws    cur_off(pc)             ;Cursor aus
  Cconws    clr_scr(pc)
* ----------------------------------------------
*        Hauptprogramm (hier wird gerechnet)
* ----------------------------------------------
  lea       x_pos(pc),a1      ;1.Position berechnen
  move.w    (x_hlb,pc),d0     ;x  Position
  add.w     (rad1,pc),d0      ;rad.1 addieren
  add.w     (rad2,pc),d0      ;rad.2 addieren
  move.w    d0,(a1)           ;sichern
  move.w    y_hlb(pc),4(a1)   ;y position
  fmove.x   #0,fp0            ;Bei 0 anfangen
  fmove.x   #1,fp1
  fdiv.w    step(pc),fp1      ;fp1=1/step
  move.w    x_hlb(pc),d0      ;d0=x_hlb
  move.w    y_hlb(pc),d1      ;d1=y_hlb
  move.w    rad1(pc),d2       ;d2=rad1
  move.w    rad2(pc),d3       ;d3=rad2
  fmove.w   freq1(pc),fp6     ;fp6=freq1
  fmove.w   freq2(pc),fp7     ;fp7=freq2
rloop:
  fmove.x   fp0,fp2           ;x=w
  fmul.x    fp6,fp2           ;x=x*f1
  fcos.x    fp2               ;x=cos(x)
  fmul.w    d2,fp2            ;x=x*rad1
  fadd.w    d0,fp2            ;x=x+x_hlb
  fmove.x   fp0,fp3           ;y=w
  fmul.x    fp6,fp3           ;y=y*freq1
```

→

```
    fmul.w    ry1(pc),fp3       ;y=y*ry1
    fsin.x    fp3               ;y=sin(y)
    fmul.w    d2,fp3            ;y=y*rad1
    fadd.w    d1,fp3            ;y=y+y_hlb
    fmove.x   fp0,fp4           ;x1=w
    fmul.x    fp7,fp4           ;x1=x1*freq2
    fcos.x    fp4               ;x1=cos(x1)
    fmul.w    d3,fp4            ;x1=x1*rad2
    fmove.x   fp0,fp5           ;y1=w
    fmul.x    fp7,fp5           ;y1=y1*freq2
    fmul.w    ry2(pc),fp5       ;y1=y1*ry2
    fsin.x    fp5               ;y1=sin(y1)
    fmul.w    d3,fp5            ;y1=y1*rad2
    fadd.x    fp4,fp2           ;x=x+x1
    fadd.x    fp5,fp3           ;y=y+y1
    bsr       line              ;linie zeichnen
    fadd.x    fp1,fp0           ;w=w+step
    fcmp.p    max(pc),fp0       ;schluß ?
    fble      rloop
    Cconin                      ;auf Taste warten
    fmovem.x  (sp)+,fp0-fp7     ;alle Register zurück
    fmovem.l  (sp)+,fpcr/fpsr/fpiar
    movem.l   (sp)+,d0-d7/a0-a6
    clr.w     -(sp)
    trap      #1                ;zurück zu GEM
* ----------------------------------------------
*             Unterprogramme
* ----------------------------------------------
* Unterprogramm für Linien
line:
    movem.l   d0-d2/a2-a3,-(sp)
    lea       a_zeiger,a3
    movea.l   (a3),a3
    lea       x_pos,a2
    move.w    #1,$18(a3)        ;farbe =1
    move.w    #-1,$20(a3)
    move.w    #$ffff,$22(a3)    ;linienstil
    clr.w     $1a(a3)           ;die restlichen
    clr.w     $1c(a3)           ;Bitplanes sind
    clr.w     $1e(a3)           ;gleich null
    fmove.w   fp2,$26(a3)       ;x neu
    fmove.w   fp3,$28(a3)       ;y neu
    move.w    (a2),$2a(a3)      ;x alt
    move.w    4(a2),$2c(a3)     ;y alt
    move.w    #0,$24(a3)        ;replace
    fmove.w   fp2,(a2)          ;neue x_pos sichern
    fmove.w   fp3,4(a2)         ;neue y_pos sichern
    aline     a_line            ;linie zeichnen
    movem.l   (sp)+,d0-d2/a2-a3
    rts
* ---------------------------------
* Unterprogramm für cardinal Eingabe
lies_card:
    movem.l   d1-d3/a0-a2,-(sp)
    movea.l   36(sp),a0         ;Bufferadr in A0
    move.w    #$1b59,(a0)       ;ESC Y in Buffer
    move.w    28(sp),2(a0)      ;Koord. in Buffer
    addi.w    #$2020,2(a0)      ;jeweils 32 addieren
    move.b    #0,4(a0)          ;Stringende
    Cconws    (a0)              ;String ausgeben
    Cconws    clr_line(pc)      ;Zeile löschen
    movea.l   32(sp),a1         ;Textadr in A1
    Cconws    (a1)              ;Text ausgeben
    move.b    31(sp),(a0)       ;Länge in Buffer
    Cconrs    (a0)              ;String einlesen
    move.b    1(a0),d1          ;Anzahl Zeichen in D1
    beq       lcl_exit          ;Anzahl=0 -> exit
    ext.w     d1
    subq.w    #1,d1
    clr.w     d2
    clr.w     d0
lc_loop:
    mulu      #10,d0            ;Zahl mal 10
    move.b    2(a0,d2.w),d3     ;Zeichen in D3
    subi.b    #$30,d3           ;ASCII - 48
    bmi       lcl_exit          ;keine Ziffer -> exit
    cmpi.b    #9,d3             ;Wert>9 ?
    bgt       lcl_exit          ;keine Ziffer -> exit
    ext.w     d3
    add.w     d3,d0             ;d0=d0+Ziffer
    addq.b    #1,d2             ;nächste Ziffer
    dbf       d1,lc_loop        ;bis zur letzten Z.
    bra       lc_ende
lcl_exit:
    moveq.l   #0,d0             ;Fehler aufgetreten
lc_ende:
    movem.l   (sp)+,d1-d3/a0-a2
    rts
* Unterprogramm für integer Eingabe
lies_int:
    movem.l   d1-d3/a0-a2,-(sp)
    movea.l   36(sp),a0         ;Bufferadr in A0
    move.w    #$1b59,(a0)       ;ESC Y in Buffer
    move.w    28(sp),2(a0)      ;Koord. in Buffer
    addi.w    #$2020,2(a0)      ;jeweils 32 addieren
    move.b    #0,4(a0)          ;Stringende
    Cconws    (a0)              ;String ausgeben
    Cconws    clr_line(pc)      ;Zeile löschen
    movea.l   32(sp),a1         ;Textadr in A1
    Cconws    (a1)              ;Text ausgeben
    move.b    31(sp),(a0)       ;Länge in Buffer
    Cconrs    (a0)              ;String eingeben
    move.b    1(a0),d1          ;Zeichen gelesen ?
    beq       lil_exit          ;nein -> exit
    ext.w     d1
    move.b    2(a0),d0          ;erstes Zeichen in D0
    clr.l     d3                ;Zahl positiv
    cmp.b     #'+',d0           ;'+' erstes Zeichen ?
    beq       li_plus
    cmp.b     #'-',d0           ;'-' erstes Zeichen ?
    beq       li_minus
    bra       li_ziffer         ;1.Zeichen ist Ziffer
li_minus:
    or.l      #$10000,d3        ;Zahl negativ
li_plus:
    moveq     #1,d2             ;1.Zeichen fällt weg
    subq.b    #2,d1             ;ein Zeichen weniger
    bra       li_pl_mi
li_ziffer:
    subq.w    #1,d1
    clr.w     d2
li_pl_mi:
    clr.w     d0
li_loop:
    mulu      #10,d0            ;Zahl mal 10
    move.b    2(a0,d2.w),d3     ;Zeichen in D3
    subi.b    #$30,d3           ;ASCII - 48
    bmi       lil_exit          ;keine Ziffer -> exit
    cmpi.b    #9,d3             ;Wert>9 ?
    bgt       lil_exit          ;keine Ziffer -> exit
    ext.w     d3
    add.w     d3,d0             ;d0=d0+Ziffer
    addq.b    #1,d2             ;nächste Ziffer
    dbf       d1,li_loop        ;bis zur letzten Z.
    swap      d3
    btst      #0,d3             ;Zahl negativ ?
    beq       li_ende           ;nein -> ende
    neg.w     d0                ;sonst negieren
    bra       li_ende           ;-> ende
lil_exit:
    moveq.l   #0,d0             ;Fehler aufgetreten
li_ende:
    movem.l   (sp)+,d1-d3/a0-a2
    rts
* ----------------------------------------------
*             Daten
* ----------------------------------------------
x_hlb:      dc.w  0
y_hlb:      dc.w  0
* Hier werden die Eingaben eingetragen
rad1:       dc.w  0             ;Radius 1
rad2:       dc.w  0             ;Radius 2
freq1:      dc.w  0             ;Frequenz 1
freq2:      dc.w  0             ;Frequenz 2
ry1:        dc.w  0             ;Konstante 1
ry2:        dc.w  0             ;Konstante 2
step:       dc.w  0             ;Genauigkeit
*
max:        dc.p  6.283185
a_zeiger:   dc.l  0
x_pos:      dc.w  0
y_pos:      dc.w  0
*
Frage1: asciiz  'Radius  1 :'
Frage2: asciiz  'Radius  2 :'
Frage3: asciiz  'Frequenz  1 :'
Frage4: asciiz  'Frequenz  2 :'
Frage5: asciiz  'Konstante 1 (1-3) :'
Frage6: asciiz  'Konstante 2 (1-3) :'
Frage7: asciiz  'Genauigkeit (>200):'
Antwort:  dc.b  0,0,0,0,0,0,0,0,0,0,0
clr_scr:  dc.b  27,'E',0
cur_off:  dc.b  27,'f',0
cur_on:   dc.b  27,'e',0
clr_line: dc.b  27,'K',0
```

Trashcan ST



Endlich: Wiederverwertbarer Müll auch bei ST

Es war einmal eine Computerfirma, die wollte einen neuen Computer vermarkten. Da dies schnell gehen sollte, versuchte man, Zeit zu sparen, wo immer es ging. Das führte dazu, daß einige Funktionen der Benutzeroberfläche, neudeutsch auch Desktop genannt, nicht durchdacht wurden.

So oder so ähnlich könnte ein Computermärchen beginnen. Aber leider war das (anscheinend) die Strategie von Atari bei der Einführung der ST-Modelle. Viele Dinge des Desktops sind nur deshalb nicht benutzerfreundlich, weil die Zeit fehlte, sie besser zu implementieren, nicht aber, weil das GEMDOS die benötigten Funktionen nicht bieten konnte.

Dazu gehört auch die Verwaltung des Papierkorbs auf dem Desktop. Wenn man einen Macintosh gewöhnt ist, ist der Doppelklick auf das Papierkorbsymbol nach dem versehentlichen Wegwerfen einer Datei nur die logische Konsequenz aus der Illusion, einen richtigen Schreibtisch vor sich zu haben. Alle, die das nicht glauben, sollen mal nach den letzten weggeworfenen Listings im heimischen Papierkorb suchen. Derjenige, der das auf einem Atari ST versucht, erntet mit einem Doppelklick auf das Papierkorbsymbol nur eine hämische Dialogbox.

Das Desktop des Atari ist nicht in der Lage, sich eine weggeworfene Datei zu „merken" und sie so für eventuelle Rettungsversuche zu konservieren. Einmal gelöscht, benötigt man Spezialprogramme und eine große Portion Glück, um zu retten, was noch zu retten ist. Diesen Umstand versucht die Gemini-Shell mit der Zweiteilung in Schredder und Papierkorb zu mildern. Dateien, die man später noch einmal retten will(...), wirft man in den Papierkorb, den man später durch das Werfen in den Schredder manuell leeren muß. Dabei ist der Papierkorb nichts anderes als ein bestimmter Ordner auf der Festplatte. Die Idee ist gar nicht schlecht, birgt aber zwei Nachteile:

1) Die Unterteilung in Papierkorb und Schredder ist natürlich nur auf dem Desktop wirksam. Wenn ein Programm eine Datei löscht, so ist diese auch weiterhin unwiederbringlich verloren.
2) Dateien, die im Papierkorb liegen, benötigen weiterhin Speicherplatz auf der Festplatte/Diskette. Wenn man erst innerhalb eines Programms merkt, daß der Platz auf dem Medium nicht reicht, kann man nur hoffen, daß das Programm eine Möglichkeit bietet, Dateien zu löschen, denn automatisch geschieht da nichts.

Dennoch, das ist der bisher einzige (mir bekannte) Ansatz, das Löschen „sicherer" zu machen. Damit ist wohl klar, welche Eigenschaften ein Programm zur Papierkorbverwaltung erfüllen muß:

Die Theorie

Zum einen muß eine automatische „Papierkorbverwaltung" Dateien sichern, die aus einem beliebigen Programm heraus gelöscht werden. Zum anderen muß es den durch diese Dateien belegten Diskettenplatz automatisch wieder freigeben, wenn er benötigt wird.

Der erste Punkt wird durch eine Änderung des *Fdelete*-Aufrufs erreicht. Dieser muß gegen eine eigene Routine ausgetauscht werden. Diese Routine löscht eine Datei gleichen Namens im Papierkorb und renamed dann die zu löschende Datei in eben diesem.

Wenn das aus irgendeinem Grund nicht möglich ist, wird die Originaldatei gelöscht. Dies läßt sich verschmerzen, da das ohnehin die Absicht des *Fdelete*-Aufrufs war. Nebenbei bemerkt folgt aus der Tatsache, daß renamed wird, auch, daß auf jedem Laufwerk ein 'Papierkorbordner' existieren muß, da das GEMDOS Dateien nicht über logische Laufwerke hinweg umbenennen kann! Da es einige (wenige) Programme gibt, die vor einer Speicheroperation den freien Speicherplatz des Laufwerks überprüfen, muß die *Dfree*-Funktion 'überarbeitet' werden. Es wird nach dem normalen Aufruf noch die Länge der im Papierkorb liegenden Dateien 'hinzugerechnet'. Dazu wird die Länge aller Dateien, die im TRASHDIR abgelegt sind, ermittelt, in Cluster umgerechnet und zur Anzahl der freien Cluster addiert.

Die Umleitung der Löschaktion ist aber nur die halbe Miete, denn auch beim Erzeugen einer Datei mittels *Fcreate* wird eine Datei mit gleichem Namen gnadenlos gelöscht. Aus diesem Grund muß auch der *Fcreate*-Aufruf dahingehend überarbeitet werden, daß eine gleichnamige Datei ins Trashdir wandert.

Dies führt aber zu einem Zwiespalt: Wenn man eine Datei im Papierkorb sichert, wenn sie mit Fcreate überschrieben werden würde, so könnte man keine Datei

mehr aus dem Papierkorb zurückholen. Jeder Versuch, diese Datei nach einer verunglückten Modifikation zu restaurieren, führt zu einem *Fcreate*-Aufruf. Dadurch würde man die zu restaurierende Datei endgültig vernichten, bevor man sie hätte retten können.

Diesem Dilemma geht mein Programm folgendermaßen aus dem Weg: Zum einen wird VOR dem *Fcreate* geprüft, ob eine Datei mit gleichem Namen existiert. Ist das nicht der Fall, wird auch die Datei im Papierkorb nicht verändert. Zum anderen verhindert das Festhalten der SHIFT-Taste eine Interpretation der Betriebssystemaufrufe durch TRASHCAN-ST. Es genügt dann, während des gesamten Kopierbzw. Rename-Vorgangs die Shift-Taste gedrückt zu halten, um eine Änderung der Dateien im Trashdir zu verhindern.

Wer möchte, kann ja mal 'Info anzeigen' im Datei-Menü anwählen. Mit gedrückter Shift-Taste und gefülltem Trashdir-Ordner sollte die Anzahl der freien Bytes deutlich kleiner sein.

Die zweite Forderung wird durch eine Änderung der *Fwrite*-Funktion erreicht. Wenn der durch die Papierkorbdateien belegte Diskettenplatz wirklich benötigt wird, müssen diese Dateien zur rechten Zeit gelöscht werden. Das ist aber relativ einfach realisierbar. Es existiert nur eine einzige Funktion, die schreibt: *Fwrite*. Es genügt, einfach den Aufruf vom *Fwrite* intern durchzuführen und dann zu überprüfen, ob die Anzahl der zu schreibenden Bytes gleich der Anzahl der geschriebenen Bytes ist. Ist das der Fall, ist alles in Ordnung; andernfalls wird die älteste Datei im Papierkorb gelöscht und dann die Schreiboperation fortgesetzt, bis entweder alle Bytes geschrieben oder aber alle Dateien im Papierkorb gelöscht wurden. Dann ist wirklich kein Platz mehr frei...

Nur leider weiß keiner, auf welchem Laufwerk gerade *Fwrite* durchgeführt wird, da das GEMDOS an dieser Stelle nur mit einem Handle, nicht aber mit dem kompletten Pfadnamen arbeitet. Aus diesem Grund werden auch noch die *Fcreate*- und *Fopen*-Aufrufe abgefangen, um in einer Tabelle die zu den Handles gehörenden Laufwerksnamen zu speichern. Wenn *Fclose* aufgerufen wird, wird das Handle auch intern wieder freigegeben.

Die Implementation

Das Programm versucht als erstes festzustellen, ob es bereits installiert war, um eine Doppelinstallation zu verhindern (die fatal wäre...). Dazu wird der GEMDOS-Vektor erfragt und mittels des XBRA-Verfahrens solange rückverfolgt, bis entweder die eigene Kennung gefunden wurde oder aber das Ende erreicht ist. Wer wissen möchte, wie das XBRA-Verfahren anzuwenden ist, der schaue sich das Listing an oder schmökere mal in [1].

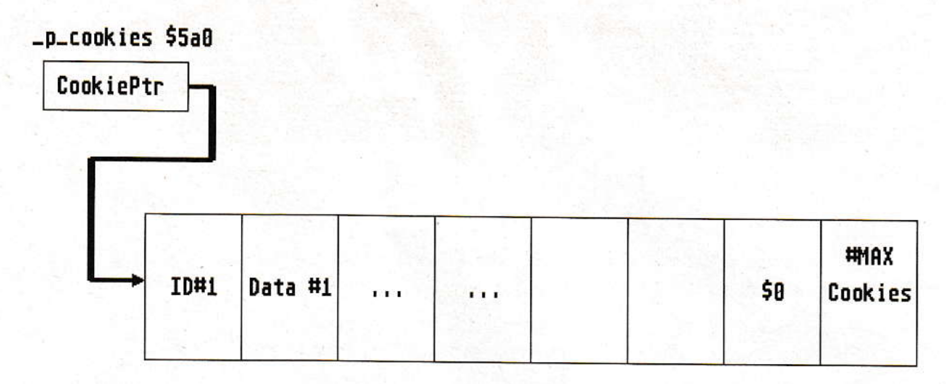


Bild 1: Cookiejar-Aufbau

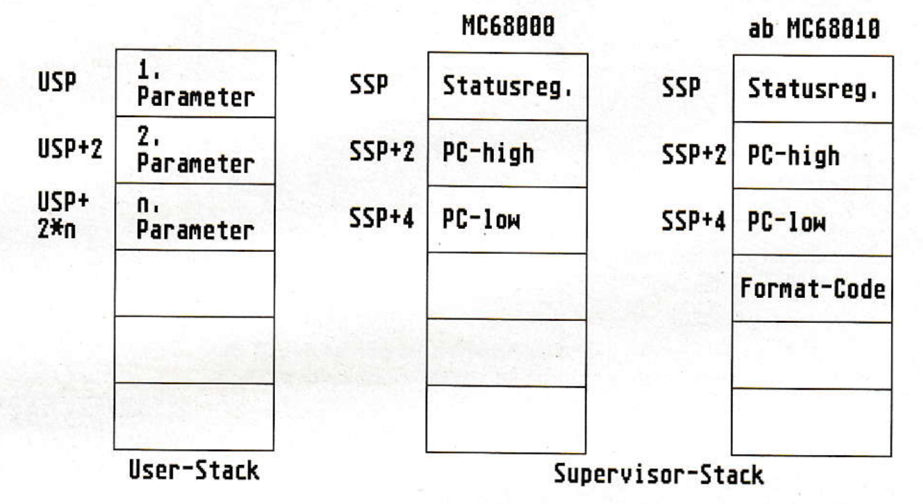


Bild 2: Stack-Aufbau im Vergleich von 68000 und 68030

Wird die eigene Kennung nicht gefunden, wird der Trap #1-Vektor auf die eigene Routine umgebogen. Danach ermittelt das Programm den verwendeten Prozessortyp. Dazu wird eine Eigenschaft der neuen TOS-Versionen (ab TOS 1.6) ausgenutzt, das Cookiejar [2]. Das Cookiejar ist nichts anderes als eine neue, von Atari dokumentierte Betriebssystemstruktur mit dem in Bild 1 beschriebenen Aufbau.

Es existiert eine neue Systemvariable, genannt *_p_cookies*, an der Adresse $5a0. Da dieser Speicherplatz auch bei allen früheren TOS-Versionen unbenutzt ist, kann man diese Struktur bei allen TOS-Versionen 'nachrüsten'. *_p_cookies* ist dann ein Pointer auf eine Liste von 8 Byte großen Elementen.

Jedes Element besteht aus:

```
dc.l „MyID"       ;Magic ID
dc.l $12345678    ;zugehörige Daten..
dc.l $0           ;Ende der Liste
dc.l $6           ;maximal mögliche Anzahl
                  der Einträge (hier: Länge
                  in Bytes: 6*8!!)
```

Es interessiert aber nur das *_CPU*-cookie, das vom TT-TOS angelegt wird. Im Datenteil ist der Prozessortyp codiert. Dabei steht eine 10 für den MC68010, eine 20 für den 68020 etc. Das ist wichtig, da der Stack-Aufbau einer Exception vom Prozessortyp abhängig ist. Ab dem MC68010 wird nämlich bei einer Exception ein zusätzliches Wort auf dem Stack abgelegt. Mit dessen Hilfe unterscheidet der Prozessor verschiedene Stack-Formate. Dieses zusätzliche Wort wird VOR den üblichen Daten (PC, Status) abgelegt und besitzt den folgenden Aufbau: %DDDD-FFFFFFFFFFFF.

Dabei codiert %FFFFFFFFFFFF die Vektornummer der auslösenden Exception. In %DDDD ist der Exception-Typ codiert. %0000 steht für eine 'normale' Exception, %1001 für einen Bus- bzw. Adreß-Error, bei dem 29 zusätzliche Wörter auf dem Stack abgelegt werden.

Da der Trap#1-Vektor ein normaler Exception-Vektor ist, wird nur das Zusatzwort gespeichert. Leider erhöht sich das Offset zu den auf dem Stack abgelegten Daten dadurch um 2 Byte, was der Grund für einige Inkompabilitäten zwischen ST und TT sein dürfte. So kommt der Traphandler nicht mehr an die Parameter, wenn man das GEMDOS aus dem

Supervisormodus heraus aufruft. Meine neue GEMDOS-Routine beachtet diesen Unterschied natürlich, um so auf jedem Atari lauffähig zu sein.

Auch ist sie wiedereintrittsfähig, was aber nicht heißen soll, daß das jetzt auch für die Originalroutinen gilt. Das GEMDOS ist nicht reentrant, aber da innerhalb des GEMDOS-Aufrufs das GEMDOS wiederum aufgerufen wird, sind die einzelnen Routinen durch Semaphoren gegen einen Wiedereintritt abgesichert. Eine Semaphore kann man sich als Schalter vorstellen, der eine Bearbeitung erlaubt oder verbietet. Dabei gilt es jedoch einige Spielregeln zu beachten. Das Abfragen und Setzen einer Semaphore muß 'unteilbar' sein, damit in Bild 2 illustrierter Zustand nicht auftreten kann.

Die Routine prüft ihre Semaphore und stellt fest, daß sie noch nicht gesetzt ist. Fein, denkt sie sich, aber während sie noch mit dem Freuen beschäftigt ist, setzt eine andere Routine (z.B. aus einem Interrupt heraus) eben diesen Schalter. Nachdem die Freude abgeklungen ist, setzt auch die ahnungslose Routine den (schon gesetzten) Schalter. Jetzt denken also beide Routinen, daß sie die exklusiven Zugriffsrechte auf die durch die Semaphore geschützten Daten haben, und legen los. Jeder male sich die Folgen aus...

Der MC68k-Befehlssatz bietet aber eine elegante Möglichkeit, dem oben beschriebenen Konflikt aus dem Weg zu gehen. Der TAS-Befehl testet die angegebene Speicherstelle, setzt die Prozessor-Flags entsprechend und schreibt dann $ff in diese Speicherstelle, ohne die Flags zu beeinflussen.

```
tas   $adr     ;Semaphore testen und
                belegen
bne   ende     ;war schon gesperrt..
clr.b $adr     ;Semaphore freigeben
ende: ..
```

Im Gegensatz zur Lösung mittels

```
tst.b $adr     ;Semaphore testen
bne ende       ;<>0 bedeutet Ende
move.b #$ff,$adr  ;Semaphore setzen..
clr.b $adr     ;Semaphore freigeben
ende: ..       ;hier geht's weiter
```

ist der erste Zyklus unteilbar. Davon wird bei allen kritischen Routinen Gebrauch gemacht.

Optionen

Kommen wir jetzt zu den Informationen, die auch den Nur-Anwender interessieren werden. Auf allen Laufwerken bzw. Disketten, auf denen Trashdir wirken soll, muß ein Ordner mit dem Namen TRASHDIR erzeugt werden. So wird es möglich, bestimmte Laufwerke/Partitionen von der automatischen Sicherung auszunehmen.

Die Dateien werden, wenn sie in das Trashdir wandern, mit dem aktuellen Datum/Uhrzeit versehen. Dadurch hat man zum einen stets die Übersicht, wann man was gelöscht hat. Zum anderen wird es dadurch erst möglich, bei Bedarf die Dateien in der Reihenfolge ihrer Löschung endgültig zu eliminieren. Dateien, die schon im Papierkorb abgelegt sind, kann man durch nochmaliges Löschen endgültig wegwerfen. Wenn man eine Datei aus dem Papierkorb zurückholen will, muß man die Shift-Taste während der gesamten Kopier- bzw. Rename-Aktion gedrückt halten. Das ist zwar nicht unbedingt nötig, aber so vermeidet man Datenverlust in der letzten Sekunde. Zur Erklärung: Wenn man eine Datei innerhalb eines Laufwerks kopiert, benötigt sie natürlich eigenen Platz auf dem Medium. Sollte der Restplatz nicht reichen, kann es passieren, daß die zu rettende Datei [die ja im TRASHDIR liegt und somit ja unnütz(!) ist] gelöscht wird, bevor sie gerettet wurde. Wenn man während dieser Zeit aber die Shift-Taste festhält, werden alle Abfragen der Mülleimer-Software umgangen (inkl. dem automatischen Löschen des Trashdirs bei Speicherplatzmangel auf dem Medium).

Beim Atari TT kann man die Trashdir-Ordner auf das Desktop ablegen. Man braucht eine Datei nur in den Originalpapierkorb zu werfen und kann es aus dem Trashdir des jeweiligen Laufwerks bequem herausholen (siehe auch Bild 3)

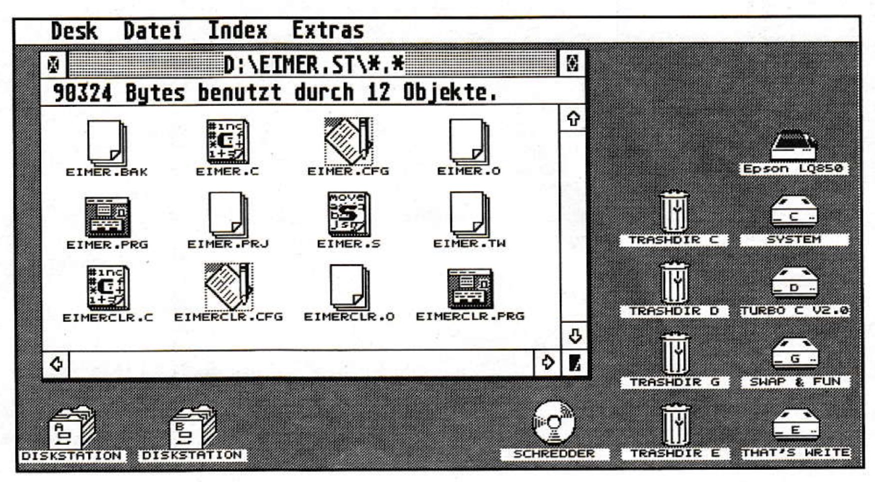


Bild 3: So könnte ein Desktop auf dem TT bzw. Mega STE aussehen.

Probleme

Probleme, die bei der Erprobung aufgefallen sind, sollen nicht unerwähnt bleiben:

Zum einen funktioniert das Programm nicht mit der komprimierenden RAM-Disk Maxidisk. Beim Schreiben auf das volle Laufwerk wird die zu schreibende Datei zerstört. Abhilfe: benutze eine andere RAM-Disk (schade).

Ein zweiter Fehler hängt mit dem TOS 1.04 zusammen. Dieses TOS kommt ins Schleudern, falls beim Umbenennen einer Datei in einen Ordner dieser erweitert werden müßte. Dieser Fall tritt nach einem Vielfachen von 32 Dateieinträgen ein (inklusive den Einträgen für aktuelles [.] und übergeordnetes [..] Directory). Wenn kein Platz auf dem Laufwerk mehr frei ist, der für den neuen Directory-Eintrag benutzt werden könnte, stürzt TOS 1.04 in diesem Fall einfach ab! Dieser Fall tritt zwar normalerweise nicht auf, regelmäßiges Leeren des Papierkorbs beugt dem aber dennoch sicher vor.

Den dritten Fehler begeht das Desktop des Atari TT (3.01 29.08.1990). Wenn es eine Datei auf ein Medium schreibt, auf das sie aber nicht vollständig paßt, so löscht es diese Datei, ohne sie vorher zu schließen. Da dies durch die Papierkorbverwaltung in einen *Frename*-Aufruf geändert wird, versucht das GEMDOS eine nicht geschlossene Datei umzubenennen. Durch diese Aktion kommt das GEMDOS so ins Schleudern, daß sowohl die Datei nicht mehr auffindbar ist, als auch der Platz auf dem Medium weiterhin belegt bleibt.

Aussichten

Für die, die alles verbessern wollen, noch eine Anregung: Das Löschen dauert umso länger, je mehr Dateien im Trashdir sind. Es wäre also ratsam, das Programm dahingehend zu erweitern, daß nach dem Start alle Dateien in den Trashdirs, die älter als z.B. eine Woche sind, gelöschet werden. Man könnte auch Dateien mit einer bestimmten Endung von der Sicherung ausnehmen... Auch könnte man die Sicherung auf bestimmte Dateiendungen beschränken... Aber das hätte das Programm noch weiter aufgebläht, als dies ohnehin schon der Fall ist.

Friedel van Megen

[1] Jankowski, Reschke, Rabich; ATARI ST Profibuch, Sybex-Verlag, ISBN 3-88745-563-0
[2] Atari STE TOS release notes, 12. Jan. 1990

```
;*******************************************
;** Mülleimer V1.1
;** Entwickelt mit MAS1.5
;** 1991 by Friedel van Megen
;**
;** (c) 1991 MAXON Computer
;*******************************************

Cconws   EQU   9  ;Das bedarf wohl keiner Erklärung...
Dgetdrv  EQU   25
Dfree EQU   54
Tgetdate EQU   42
Tgettime EQU   44
Fdatime  EQU   87
Fgetdta  EQU   47
Fsetdta  EQU   26
Fdelete  EQU   65
Fwrite   EQU   64
Frename  EQU   86
Fsfirst  EQU   78
Fsnext   EQU   79
Fcreate  EQU   60
Fopen EQU   61
Fclose   EQU   62
Ptermres EQU 49
gemdos   EQU   1

Setexec  EQU   5
bios  EQU   13

Supexec  EQU   38
xbios EQU   14

;*******************************************
;** Ab hier wird es interessant
;*******************************************
    TEXT
tr_start:   pea   myname
    move.w   #Cconws,-(sp)
    trap #gemdos
    addq.l   #6,sp

    move.l   #-1,-(sp)
    move.w   #33,-(sp)   ;gemdos-vector-number
    move.w   #Setexec,-(sp)
    trap  #bios
    addq.l   #8,sp
    move.l   d0,a0

in_test: cmp.l #0,a0
    beq   install     ;Kettenende erreicht
    cmp.l #'XBRA',-12(a0)
    bne   install
    cmp.l #'FGEM',-8(a0) ;eigenen ID suchen
    beq   no_inst
    move.l   -4(a0),a0   ;nächstes Kettenglied

    bra   in_test

no_inst: pea   warschon    ;das war wohl nichts

    move.w   #Cconws,-(sp)
    trap  #gemdos
    addq.l   #6,sp
    clr   -(sp)
    trap  #gemdos

install: pea   testST
    move.w   #Supexec,-(sp)
    trap  #xbios
    addq.l   #6,sp
    pea   n_trp1      ;GEMDOS patchen
    move.w   #33,-(sp)   ;gemdos-vector-number
    move.w   #Setexec,-(sp)
    trap  #bios
    addq.l   #8,sp
    move.l   d0,sv_trp1  ;alten Vektor sichern

    move.l   4(sp),a1    ;Basepage Pointer vom Stack

    move.l   #$100,a0
    add.l 12(a1),a0
    add.l 20(a1),a0
    add.l 28(a1),a0
    move.w   #0,-(sp)    ;wir bleiben resident!
    move.l   a0,-(sp)    ;Programmlänge
    move.w   #Ptermres,-(sp)
    trap  #gemdos     ;Und Schluß...

    DATA
myname:  dc.b  "Mülleimer V1.1 (vom 11. Mai 1991)",10,13
    dc.b  "(P) 1991 Friedel van Megen",10,13,0
warschon:   dc.b  "Der Mülleimer war schon installiert",10,13,0

    even
    TEXT

;*******************************************
;** Test auf den Prozessor
;*******************************************
testST:  move.l   $5a0,d0     ;cookie-root-pointer
    beq   endST
    move.l   d0,a0

    move.l   #'_CPU',d0
testST1: move.l   (a0)+,d1
    beq   endST
    cmp.l d0,d1
    beq   found    ;_CPU cookie gefunden
    addq.l   #4,a0
    bra   testST1

found:   move.l   (a0),d1
 ;Prozessortyp holen (0, 10, 20 steht für MC68000, MC68010...)
    beq   endST    ;nur ein MC68000

    add.w #2,stk_offset

endST:   move.l   $4f2,a0     ;Sysbase
    move.w   2(a0),d0
    cmp.w #$0100,d0
    bne   testST2
    rts

testST2: move.l   36(a0),shift   ;Pointer auf SHIFT-Status holen (ab TOS1.2)
    rts

;*******************************************
;** modifizierter TRAP #1 Handler, XBRA-tauglich
;*******************************************
    SUPER
    dc.l  'XBRA'
    dc.l  'FGEM'
sv_trpl: dc.l  0     ;savearea für gemdos vektor

n_trp1:  tst.w sema  ;darf ich was machen?
    bne   end_trp1    ;JA ->

    move.l   shift,a0    ;falls SHIFT gedrückt wurde: ABBRUCH
    move.b   (a0),d0
    and.b #%11,d0
    bne   end_trp1

    move.l   a7,a0    ;Stackpointer bestimmen
    move.w   stk_offset,d0
    lea   6(a0,d0.w),a0  ;Offsetänderung ab MC68010 ausgleichen
    move.w   (sp),d0  ;Statuspaket, das beim TRAP abgelegt wurde
    btst  #13,d0
    bne   in_supm     ;ok, Supervisor
    move.l   usp,a0      ;Aufruf aus USER-Mode
in_supm: move.w   (a0)+,d0 ;Funktionscode
    cmp.w #Fdelete,d0 ;Fdelete?
    beq   myFdel
    cmp.w #Fwrite,d0  ;Fwrite?
    beq   myFwrite
    cmp.w #Fclose,d0  ;Fclose
    beq   myFclose
    cmp.w #Fcreate,d0 ;Fcreate
    beq   myFcreat
    cmp.w #Fopen,d0   ;Fopen
    beq   myFopen
    cmp.w #Dfree,d0   ;Dfree
    beq   myDfree
```

→

```

end_trpl:   move.l   sv_trpl,a0
    jmp   (a0)      ;dann eben nicht....
    USER

;*******************************************
;** neue Dfree-Routine; pointer auf Parameter in A0
;*******************************************
    SUPER
myDfree: tas   sema  ;sicherstellen, daß wirklich niemand
    beq   dfreel      ;sonst dazwischen funkt
    move.l   sv_trpl,a0
    jmp   (a0)

dfreel:  movem.l  d3/d4/a3/a4,-(sp)
    move.l   a0,a3
    move.l   (a0),a4       ;Pointer auf DISKINFO
    move.w   #Fgetdta,-(sp) ;alten DTA sichern
    bsr   _gemdos
    addq.l   #2,sp
    move.l   d0,sv_dta
    pea   my_dta       ;und eigenen setzen
    move.w   #Fsetdta,-(sp)
    bsr   _gemdos
    addq.l   #6,sp

    move.w   4(a3),-(sp) ;Originalroutine ausführen
    move.l   (a3),-(sp)
    move.w   #Dfree,-(sp)
    bsr   _gemdos
    addq.l   #8,sp
    move.l   d0,-(sp)      ;Status DFREE sichern
    tst.w d0
    bmi   endDfree

    lea   trashname,a0    ;ist Laufwerk angegeben
    move.b   #'*',0(a0)
    move.b   #'.',1(a0)
    move.b   #'*',2(a0)
    clr.b 3(a0)
    lea   trashdir,a0
    tst.w 4(a3)
    beq   dfree2

    move.b   5(a3),trashdrive
    add.b #'A' - 1,trashdrive   ;Laufwerk eintragen
    lea   trashdrive,a0

dfree2:  move.w   #%100110,-(sp) ;WRI/SYS/HID-Bits
    move.l   a0,-(sp)
    move.w   #Fsfirst,-(sp)
    bsr   _gemdos
    addq.l   #8,sp
    tst.l d0
    bmi   endDfree      ;Keine Datei, bzw kein Ordner vorhanden

    lea   my_dta,a3
    move.l   8(a4),d4
    mulu.w   14(a4),d4       ;Anzahl der Bytes pro cluster
    tst.w d4
    beq   endDfree     ;Unsinn im Parameterblock

    move.l   26(a3),d3       ;Programmlänge
    add.l d4,d3       ;in Cluster umrechnen
    subq.l   #1,d3
    divu  d4,d3
    and.l #$ffff,d3

dfree3:  move.w   #Fsnext,-(sp)
    bsr   _gemdos
    addq.l   #2,sp
    tst.l d0
    bmi   dfree4       ;Ende der Verzeichniseinträge erreicht
    move.l   26(a3),d0       ;Programmlänge
    add.l d4,d0
    subq.l   #1,d0
    divu  d4,d0
    and.l #$ffff,d0        ;Der Rest interessiert nicht
    add.l d0,d3     ;und addieren...
    bra   dfree3

dfree4:  add.l d3,0(a4)      ;es ist noch mehr frei als Du denktst...

endDfree:   move.l   sv_dta,-(sp)    ;alten DTA restaurieren
    move.w   #Fsetdta,-(sp)
    bsr   _gemdos
    addq.l   #6,sp
    move.l   (sp)+,d0     ;Status DFREE restaurieren
    movem.l  (sp)+,d3/d4/a3/a4
    clr.w sema
    rte
    USER

;*******************************************
;** neue Fclose-routine; pointer auf parameter in A0
;*******************************************
    SUPER
myFclose:   tas   sema      ;sicherstellen, daß wirklich niemand
    beq   fclosel    ;sonst dazwischen funkt
    move.l   sv_trpl,a0
    jmp   (a0)

fclosel: move.w   (a0),-(sp)   ;sichern
    move.w   (a0),-(sp)
    move.w   #Fclose,-(sp)
    bsr   _gemdos
    addq.l   #4,sp
    tst.l d0
    bmi   endclose
    lea   hdl_tab,a0
    move.l   d1,-(sp)     ;d1 sichern
    move.w   4(sp),d1
    bmi   fclose2
    clr.b 0(a0,d1.w)  ;Eintrag löschen
fclose2: move.l   (sp)+,d1

endclose:   move.w   (sp)+,a0
    clr.w sema
    rte
    USER

;*******************************************
;** neue Fcreate-routine; pointer auf parameter in A0
;*******************************************
    SUPER
myFcreat:   move.w   #Fcreate,d0
    tas   sema      ;sicherstellen, daß wirklich niemand
    beq   fcreal   ;sonst dazwischen funkt
    move.l   sv_trpl,a0
    jmp   (a0)

fcreal:  movem.l  a1-a3/d1,-(sp)
    move.w   d0,-(sp)      ;Opcode retten!
    move.l   a0,a1       ;Sichern vom Pointer auf Pointer auf den Dateinamen
    move.w   #Dgetdrv,-(sp) ;aktuelles Laufwerk bestimmen
    bsr   _gemdos
    addq.l   #2,sp
    add.w #'A',d0       ;Laufwerksname
    move.l   d0,d1

    move.l   (a1),a0       ;Pointer auf Dateiname
    move.b   1(a0),d0      ;Pfad prüfen
    cmp.b #':',d0      ;ex Laufwerksangabe ?
    bne   fcrea2       ;NEIN ->
    move.b   (a0),d1      ;sonst übernehmen

fcrea2:  move.w   (sp),d0   ;NICHT (sp)+ !
    cmp.w #Fcreate,d0
    bne   fcrea3       ;nur bei Fcreate
    move.b   d1,trashdrive  ;Laufwerk eintragen
    move.l   a0,a2     ;Default
fcrea21: move.b   (a0)+,d0
                              ;Programmnamen suchen
```

→

```
    beq   fcrea23    ;Ende erreicht
    cmp.b #'\',d0
    beq   fcrea22    ;'\', oder ':' sind
                      "Trenner"
    cmp.b #':',d0
    bne   fcrea21
fcrea22: move.l   a0,a2     ;Pointer merken
    bra   fcrea21

fcrea23: lea   trashname,a0
                  ;Programmnamen übertragen
fcrea24: move.b   (a2)+,(a0)
    tst.b (a0)+
    bne   fcrea24

    move.l    (a1),a0      ;Pointer auf
                            Dateinamen
    bsr   fexist
    tst.l d0
    bmi   fcrea25       ;nur löschen, wenn
                         anzulegende Datei ex.
    move.l    #trashdrive,-(sp) ;Duplikat im
                            Mülleimer löschen
    move.w    #Fdelete,-(sp)
    bsr   _gemdos
    addq.l    #6,sp

fcrea25: move.l   #trashdrive,-(sp) ;Datei
                      in den Müll schieben
    move.l    (a1),-(sp)  ;Pointer auf
                            Dateiname
    clr.w -(sp)
    move.w    #Frename,-(sp)
    bsr   _gemdos
    lea   12(sp),sp
    tst.l d0
    bne   fcrea3        ;Fehler beim renamen

    move.l    #trashdrive,a3
    bsr   tim_upd       ;Zeit updaten

fcrea3:  move.w   (sp),d0       ;Opcode
                                zurückholen
    move.w    4(a1),-(sp)
    move.l    (a1),-(sp)
    move.w    d0,-(sp)      ;Fcreate/Fopen ist
                              identisch
    bsr   _gemdos
    addq.l    #8,sp
    tst.w d0    ;nur positive Handles
                 zulassen
    bpl   fcrea4        ;kein Fehler beim
                         öffnen der Datei

    cmp.l #-36,d0       ;Kein Directoryplatz
                         mehr frei
    bne   endcreat

    bsr   trashclr    ;Datei löschen
    tst.l d0
    beq   endcreat    ;Trashcar war leer
    move.w    (sp),d0      ;Opcode zurückholen
    move.w    4(a1),-(sp)
    move.l    (a1),-(sp)
    move.w    d0,-(sp)      ;Fcreate/Fopen ist
                              identisch
    bsr   _gemdos
    addq.l    #8,sp
    tst.w d0    ;Fehler beim Öffnen
    bmi   endcreat

fcrea4:  lea    hdl_tab,a0
    move.b    d1,0(a0,d0.w)    ;Laufwerk
                                eintragen

endcreat:   addq.l    #2,sp
                            ;Stackkorektur
    movem.l  (sp)+,a1-a3/d1
    clr.w sema
    rte
    USER

;*******************************************
;** Gemdos aufrufen (anspringen mittels
    bsr _gemdos !!!)
;*******************************************
    SUPER
_gemdos: move.w   sr,d0     ;Prozessor
                        Exception vorgaukeln
    tst.w stk_offset
    beq   _gemdos1

    move.l    (sp)+,a0
    move.w    #33,-(sp)         ;Zusätzliches
                   Wort ab MC68010 (siehe Text)
    move.l    a0,-(sp)

_gemdos1:    move.w    d0,-(sp)
    move.l    sv_trp1,a0
    jmp    (a0)      ;GEMDOS aufrufen
    USER

;*****************************************
;** Test auf Existenz einer Datei,
    Eingabe: Pointer auf Dateiname(A0)
;**
    Ausgabe: <0 Datei ex. NICHT
;*****************************************
fexist:  movem.l d3/a1,-(sp)
    clr.l d3     ;Fehler (default)
    move.l    a0,a1
    move.w    #Fgetdta,-(sp) ;alten DTA
                               sichern
    bsr   _gemdos
    addq.l    #2,sp
    move.l    d0,sv_dta
    pea    my_dta        ;und eigenen setzen
    move.w    #Fsetdta,-(sp)
    bsr   _gemdos
    addq.l    #6,sp

    move.w    #%100110,-(sp) ;WRI/SYS/HID-
                               Bits
    move.l    a1,-(sp)
    move.w    #Fsfirst,-(sp)
    bsr   _gemdos
    addq.l    #8,sp
    move.l    d0,d3     ;status merken
    tst.l d0
    bmi    endex    ;Die Datei ex. nicht
    moveq.l  #0,d3

endex:    move.l    sv_dta,-(sp)    ;alten
                       DTA restaurieren
    move.w    #Fsetdta,-(sp)
    bsr   _gemdos
    addq.l    #6,sp
    move.l    d3,d0     ;Status zrückgeben
    movem.l   (sp)+,d3/a1
    rts

;******************************************
;** Zeit/Datum einer Datei updaten
    Eingabe: A3
;******************************************
tim_upd: move.w   #0,-(sp)      ;Versuche
              die Datei im Müll zu öffnen
    move.l    a3,-(sp)
    move.w    #Fopen,-(sp)
    bsr   _gemdos
    addq.l    #8,sp
    tst.l d0
    bmi   end_upd      ;hat nicht geklappt
    move.w    d0,-(sp)     ;Fclose vorbereiten
    move.w    #Fclose,-(sp)

    move.w    #1,-(sp)      ;Tdatime
                             vorbereiten
    move.w    d0,-(sp)
    pea    timeptr
    move.w    #Fdatime,-(sp)

    move.w    #Tgettime,-(sp)   ;Zeit holen
    bsr   _gemdos
    addq.l    #2,sp
    move.w    d0,timeptr
    move.w    #Tgetdate,-(sp)   ;Datum holen
    bsr   _gemdos
    addq.l    #2,sp
    move.w    d0,timeptr+2

    bsr   _gemdos     ;Fdatime
    lea   10(sp),sp

    bsr   _gemdos     ;Fclose
    addq.l    #4,sp
end_upd: rts
```

→

```
;*******************************************
;** neue Fopen-routine; pointer auf
     parameter in A0
;*******************************************
myFopen: move.w  #Fopen,d0
    tas   sema     ;sicherstellen, daß
                   wirklich niemand
    beq   fcrea1      ;sonst dazwischen
                      funkt
    move.l   sv_trp1,a0
    jmp   (a0)

;*******************************************
;** neue Fdelete-routine; pointer auf
     parameter in A0
;*******************************************
    SUPER
myFdel:  tas   sema         ;sicherstellen,
                     daß wirklich niemand
    beq   fdel1   ;sonst dazwischen funkt
fdelerr: move.l   sv_trp1,a0
    jmp   (a0)

fdel1:   movem.l  a1-a4,-(sp)
    move.l   (a0),a4
    move.l   (a0),a1      ;Pointer auf zu
                   löschenden Programmnamen
    move.l   a1,a2      ;Pointer sichern...
    move.b   1(a1),d0
    clr.b trashdrive   ;keine
        Laufwerksangabe (default benutzen)
    cmp.b #':',d0
    bne   fdel2
    move.b   (a1),trashdrive
                 ;Laufwerksnamen eintragen

fdel2:   move.b    (a1)+,d0
                     ;Programmnamen suchen
    beq   fdel4     ;Ende erreicht
    cmp.b #'\',d0
    beq   fdel3     ;'\', oder ':' sind
                     "Trenner"
    cmp.b #':',d0
    bne   fdel2
fdel3:   move.l    a1,a2     ;Pointer merken
    bra   fdel2

fdel4:   lea   trashname,a1
                 ;Programmnamen übertragen
fdel41:  move.b   (a2)+,(a1)
    tst.b (a1)+
    bne   fdel41

    lea   trashdrive,a3 ;Wenn eine
     Laufwerksangabe vorhanden, benutzen...
    tst.b trashdrive
    bne   fdel5
    lea   trashdir,a3

fdel5:   move.l   a3,-(sp)     ;Duplikat im
                         Mülleimer löschen
    move.w   #Fdelete,-(sp)
    bsr   _gemdos
    addq.l   #6,sp
    move.l   d0,-(sp)     ;Status merken
    move.l   a3,-(sp)     ;Datei in den Müll
                         schieben
    move.l   a4,-(sp)
    clr.w -(sp)
    move.w   #Frename,-(sp)
    bsr   _gemdos
    lea   12(sp),sp
    tst.l d0
    bne   fdel6

    bsr   tim_upd      ;Zeit updaten
    bra   fdel_end

fdel6:   cmp.l #-34,d0        ;Ordner nicht
                        gefunden
    bne   fdel_end        ;alles klar
    move.l   a4,-(sp)     ;Dann eben das
                         Original löschen
    move.w   #Fdelete,-(sp)
    bsr   _gemdos
    addq.l   #6,sp

fdel_end:   tst.l d0
    bmi   fdel_e2     ;Fehler beim löschen
    addq.l   #4,sp     ;Status vergessen
    bra   fdel_e4

fdel_e2: tst.l (sp)      ;konnte denn die
          Datei im Müll gelöscht werden?
    bmi   fdel_e3     ;NEIN ->
    move.l   (sp)+,d0     ;alten Status
                            nehmen
    bra   fdel_e4

fdel_e3: addq.l   #4,sp  ;nicht Müll und
                           nicht vorhanden

fdel_e4: movem.l  (sp)+,a1-a4
    clr.w sema       ;Semaphore freigeben
    rte
    USER

    DATA
timeptr: dc.w  0,0,0,0
    TEXT

;*******************************************
;** neue Fwrite-routine
;*******************************************
    SUPER
myFwrite:   tas   sema      ;sicherstellen,
                   daß wirklich niemand
    beq   fwri1   ;sonst dazwischen funkt
    move.l   sv_trp1,a0
    jmp   (a0)

fwri1:   movem.l  a3/a4/d3,-(sp)
    move.l   2(a0),d3      ;COUNT
    move.l   6(a0),a3      ;BUFFER
    move.l   a0,a4     ;Pointer auf Paramter

fwri2:   move.l   a3,-(sp)      ;erst
      versuchen alles normal zu schreiben
    move.l   d3,-(sp)
    move.w   (a4),-(sp)
    move.w   #Fwrite,-(sp)
    bsr   _gemdos
    lea   12(sp),sp
    tst.l d0
    bmi   fwri_end     ;Fehler, den ich
                    NICHT beheben kann...
    cmp.l d0,d3
    bne   fwri3      ;Platz hat noch nicht
                      gereicht
    move.l   2(a4),d0
    bra   fwri_end     ;Zurückgeben der
                     geschriebenen Bytes

fwri3:   add.l d0,a3
    sub.l d0,d3       ;der Rest muß noch
                    geschrieben werden
    bsr   trashclr
    tst.l d0
    bne   fwri2       ;Der Trashcan war noch
                    nicht leer...
    move.l   2(a4),d0
    sub.l d3,d0    ;Wahre Anzahl
           geschriebener Bytes berechnen

fwri_end:   movem.l  (sp)+,a3/a4/d3
    clr.w sema       ;Semaphore freigeben
    rte
    USER

trashclr:   movem.l  d1/d3/a1-a3,-(sp)
    clr.l d3     ;Fehler (default)
    move.w   #Fgetdta,-(sp) ;alten DTA
                            sichern
    bsr   _gemdos
    addq.l   #2,sp
    move.l   d0,sv_dta
    pea   my_dta        ;und eigenen setzen
    move.w   #Fsetdta,-(sp)
    bsr   _gemdos
    addq.l   #6,sp

    lea   hdl_tab,a0  ;Laufwerk holen
    clr.b trashdrive
    move.w   (a4),d0        ;Handlenummer holen
    bmi   trash1
    move.b   0(a0,d0.w),trashdrive

trash1:  lea   trashdrive,a0
         ;Laufwerksbezeichnung vorhanden?
```

→

```
    tst.b trashdrive
    bne   trash2      ;JA ->
    lea   trashdir,a0

trash2:  lea   trashname,a1   ;nach löschbaren Dateien suchen
    move.b   #'*',(a1)
    move.b   #'.',1(a1)
    move.b   #'*',2(a1)
    clr.b 3(a1)
    move.w   #%100110,-(sp) ;WRI/SYS/HID-Bits
    move.l   a0,-(sp)
    move.w   #Fsfirst,-(sp)
    bsr   _gemdos
    addq.l   #8,sp
    tst.l d0
    bmi   endtrash    ;Keine Datei, bzw. kein Ordner vorhanden

    lea   my_dta,a3
    lea   tname,a0
    lea   30(a3),a2      ;Pointer auf Dateinamen
    moveq.l  #14,d0
tloop1:  move.b   (a2)+,(a0)+
    dbra  d0,tloop1      ;Dateiname übertragen
    move.w   24(a3),tdate   ;älteste Datei suchen (Pointer auf Datum)
    move.w   22(a3),ttime

trash3:  move.w   #Fsnext,-(sp)
    bsr   _gemdos
    addq.l   #2,sp
    tst.l d0
    bmi   trash4      ;Ende der Verzeichniseinträge erreicht

    move.l   22(a3),d0      ;Alter vergleichen
    swap  d0    ;Zeit/Datum vertauschen
    move.l   tdate,d1
    cmp.l d1,d0
    bpl   trash3      ;die neue Datei ist wohl älter

    lea   tname,a0    ;Die Datei ist wirklich älter...
    lea   30(a3),a2   ;Pointer auf Dateinamen
    moveq.l  #14,d0
tloop2:  move.b   (a2)+,(a0)+
    dbra  d0,tloop2      ;Dateiname übertragen
    move.w   24(a3),tdate
    move.w   22(a3),ttime
    bra   trash3

trash4:  lea   tname,a0
    lea   trashname,a2   ;Pointer auf Dateinamen
    moveq.l  #14,d0
tloop3:  move.b   (a0)+,(a2)+
    dbra  d0,tloop3      ;Dateiname übertragen
    lea   trashdrive,a0
     ;Laufwerksbezeichnung vorhanden?
    tst.b trashdrive
    bne   trash5      ;JA ->
    lea   trashdir,a0

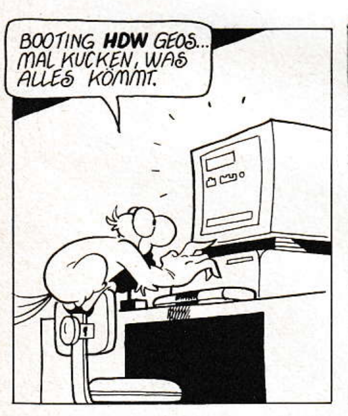
trash5:  move.l   a0,-(sp)
    move.w   #Fdelete,-(sp)
    bsr   _gemdos
    addq.l   #6,sp
    tst.l d0    ;Fehler beim Löschen?
    bmi   endtrash    ;JA ->
    moveq.l  #1,d3    ;älteste Datei gelöscht

endtrash:   move.l   sv_dta,-(sp)   ;alten DTA restaurieren
    move.w   #Fsetdta,-(sp)
    bsr   _gemdos
    addq.l   #6,sp
    move.l   d3,d0    ;Status zrückgeben
    movem.l  (sp)+,d1/d3/a1-a3
    rts

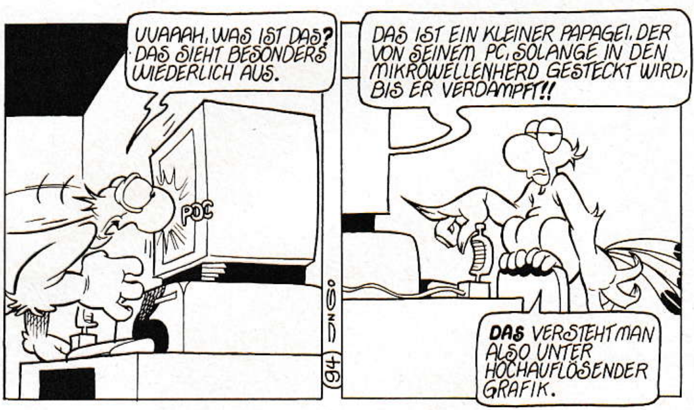
    DATA
tdate:   dc.w  0
ttime:   dc.w  0
tname:   ds.b  20
    TEXT

;*******************************************
;** datasegment
;*******************************************
      DATA
sema:    dc.w    0         ;eine Semaphore
shift:       dc.l    $e1b  ;Shift-Status (Hier noch für TOS1.0)
stk_offset: dc.w  0      ;Stackoffset beim MC68000: 0, (ab 68010: 2)
sv_dta:      dc.l   0
my_dta:      ds.b   64     ;Puffer für den DTA
trashdrive: dc.b  "X:"
trashdir:       dc.b    "\TRASHDIR\"
trashname:  ds.b  18
hdl_tab:     ds.b   128       ;Zuordnung handle <-> Laufwerk
      END
```

ROCKUS



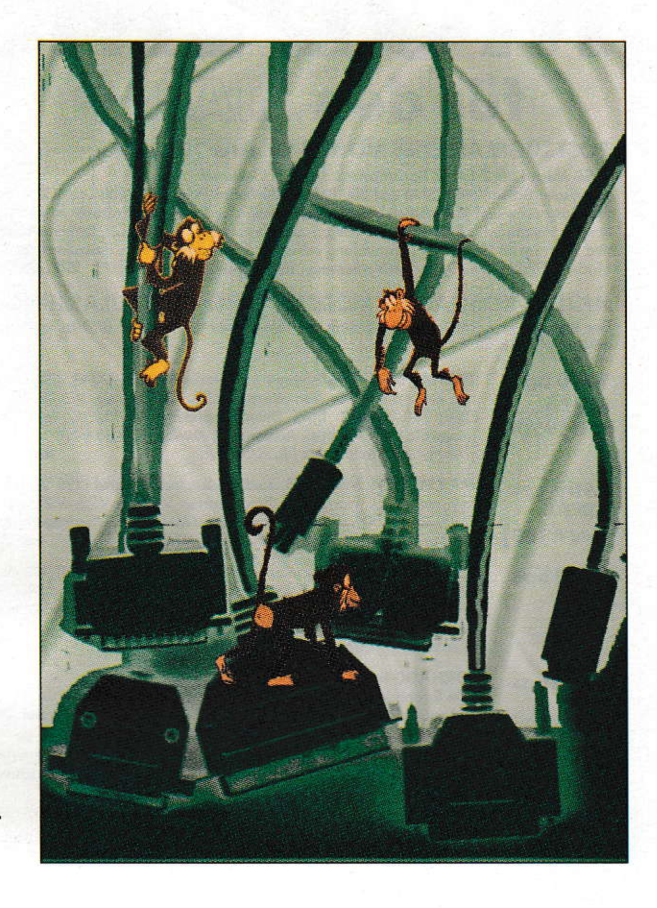




Schnittstellen-Dschungel

Neue Rechner - Neue Schnittstellen

Teil 4



Heute folgt endlich der letzte Teil der Listings, der das in Teil 1 angekündigte Terminalprogramm vervollständigt. Da das Programm regen Gebrauch von Fenstern macht, wollen wir hier nicht nur das Programm beschreiben, sondern die Gelegenheit nutzen, und uns zusätzlich näher mit einem Teil der Mensch-Maschine-Schnittstelle beschäftigen, nämlich der oft vernachlässigten Fensterprogrammierung.

Unserer Ansicht nach ist nämlich mittlerweile über die Programmierung von Dialogboxen und Menüs schon ziemlich viel geschrieben worden, so daß die meisten Programmierer problemlos in der Lage sind, GEM-Programme zu entwickeln, die diese Objekte benutzen. Das zeigt auch ein Blick auf viele PD-Programme, die zwar Dialoge und Menüs verwenden, ansonsten aber weder Fenster noch ein eigenes Desktop benutzen. Offenbar herrscht in dieser Beziehung noch ein Nachholbedürfnis, dem wir hiermit nachkommen wollen.

Daher beschreiben wir im folgenden nicht nur „graue Theorie“ mit Funktionsfragmenten, die niemand zu einem lauffähigen Programm zusammenstellen kann, sondern wir beziehen uns dabei auf unsere Listings. So kann das Zusammenwirken der verschiedenen Komponenten genau untersucht werden, was mit einzelnen Bruchstücken nicht möglich wäre. Jeder kennt die üblichen GEM-Kurse, deren einzelne Listings zwar funktionieren, jedoch im Zusammenspiel mit anderen Komponenten oftmals unerwartete Ergebnisse hervorrufen oder in der veröffentlichten Form für eigene Zwecke meist nicht zu gebrauchen sind. Zudem beginnen die meisten GEM-Kurse mit den Grundlagen, verschweigen jedoch die komplizierteren, aber dadurch auch interessanten Details, die sich der Programmierer dann in mühevoller Kleinarbeit selbst erarbeiten muß. Wir setzen deshalb die Grundlagen (C-Programmierung und GEM-Grundkenntnisse) voraus und fangen hoffentlich da an, wo andere Kurse aufhören. Es läßt sich allerdings nicht vermeiden, daß wir dabei ganz vorne beginnen.

Am Anfang war das Fenster...

Beginnen wir also mit der Fensterprogrammierung. Der Anfang aller Dinge ist hierbei die Funktion *wind_create()*. Wir wollen hier nicht jeden einzelnen Parameter bis ins Detail beschreiben, das kann in jedem Handbuch nachgelesen werden. Vielmehr wollen wir die Zusammenhänge verdeutlichen, die zwischen den einzelnen Funktionen bestehen, um die Vorgehensweise bei der Fensterprogrammierung besser verständlich zu machen.

Wie wohl bekannt sein dürfte, muß vor dem Öffnen eines Fensters dieses erst einmal mittels oben genannter Funktion erzeugt werden. *wind_create()* legt dazu im Speicher eine Datenstruktur an, die das Fenster beschreibt, und liefert ein Handle zurück, mit dem das Fenster in Zukunft angesprochen wird. Unter anderem verlangt die Funktion als Parameter die Position und die maximale Größe des Fensters. Wie man jedoch bei einer späteren Größenänderung des Fensters feststellen kann, ist es ohne weiteres möglich, das Fenster beliebig groß zu machen, sei es nun mit *wind_open()* oder durch Aufziehen des Fensters mit der Maus. Wer sich nun fragt, wozu *wind_create()* die Maximalgröße eigentlich wissen will, fragt sich dies sicherlich nicht ganz zu Unrecht.

Vom Window verweht

Zuerst einmal: es ist tatsächlich möglich, dem Fenster jegliche beliebige Größe zu geben, auch wenn sie die bei *wind_create()* angegebene Maximalgröße überschreitet. Diese Maximalgröße kann jedoch zu einem späteren Zeitpunkt (z.B. bei Anklikken der Fullbox des Fensters) mit *wind_get(..., WF_FULLXYWH,...)* wieder abgefragt werden, um das Fenster auf die vorher festgelegte Maximalgröße zu setzen. Man kann jedoch auch auf die hier beschriebene Möglichkeit verzichten und die Maximalgröße bei jedem Anklicken der Fullbox neu berechnen. Schließlich kann es auch vorkommen, daß die spätere Maximalgröße des Fensters bei seiner Erzeugung noch gar nicht bekannt ist. Das ist in unserem Terminalprogramm zwar nicht der Fall, trotzdem wird die Maximalgröße hier zum Beispiel über ein Flag verwaltet (siehe Funktion *wm_fulled()* im Listing WINDOW.C).

Soll das Fenster auf die größtmögliche Größe, d.h. in der Regel Bildschirmgröße, gesetzt werden, so kann es nützlich sein, die Größe des Desktops abzufragen. Dazu muß man wissen, daß das Desktop **immer** das Fenster-Handle 0 besitzt. Mittels *wind_get(0,WF_WORKXYWH,...)* kann nun die Größe des Desktops festgestellt werden.

Jetzt ist der Zeitpunkt gekommen, Fenstertitel sowie Infozeile (sofern vorhanden) mittels *wind_set()* zu setzen. Dies muß unbedingt **vor** dem Öffnen des Fensters geschehen, damit die entsprechenden Fensterkomponenten einen definierten Inhalt haben. Ansonsten kann es passieren, daß der Rechner beim Öffnen des Fensters abstürzt. Zu beachten ist ferner, daß die übergebenen Zeichenketten in einem statischen Speicherbereich liegen müssen, denn das AES greift bei einem Redraw immer direkt auf die vom Programmierer übergebenen Adressen zu!

Jetzt könnten wir das Fenster eigentlich an jeder beliebigen Stelle des Bildschirms öffnen - doch wer will das schon? Mit einem einfachen Trick läßt sich zumindest bei der Ausgabe von Text und bei der Benutzung von *vro_cpyfm()* die Arbeitsgeschwindigkeit erheblich beschleunigen. Dazu muß lediglich die x-Koordinate des Arbeitsbereiches unseres Fensters auf eine Zeichengrenze gelegt werden. Da jedoch bei dessen Öffnen die Abmessungen des gesamten Fensters angegeben werden müssen, ist zunächst mittels *wind_calc(WC_BORDER,...)* von der Größe des Arbeitsbereiches auf die Gesamtgröße des Fensters umzurechnen. Hierbei ist für die x-Koordinate des Arbeitsbereiches auch die Null zulässig, so daß der ein Pixel breite linke Rand des Fensters außerhalb des Desktops liegt.

Fenster zum Hof

Nach dem Initialisieren des Fensters wird es jetzt endlich mit *wind_open()* geöffnet. Als Parameter werden nur die Anfangsposition und -größe übergeben. Mancher wird sich jetzt vielleicht fragen, warum *wind_create()* und *wind_open()* nicht in einem Betriebssystemaufruf zusammengefaßt worden sind. Die Antwort ist jedoch ganz einfach: Durch die zwei getrennten Aufrufe können beispielsweise **unbedingt** benötigte Fenster zu Beginn des Programmes erzeugt werden; und falls nicht genügend Fenster erzeugt werden konnten, läßt sich rechtzeitig eine Fehlermeldung generieren. Die Fenster sind somit reserviert und bei Bedarf zu öffnen, ohne den Fall betrachten zu müssen, daß zu einem kritischen Zeitpunkt keine Fenster mehr verfügbar sind. Ein solches Vorgehen sollte jedoch nur dann in Betracht gezogen werden, wenn das entsprechende Programm ohne eine Mindestanzahl von Fenstern nicht auskommt. Ansonsten würden nur unnötig Fenster-Handles belegt, die beispielsweise auch von Accessories benutzt werden. Schließlich ist der Atari bekanntermaßen nur mit acht Fenstern gesegnet, von denen das Desktop bereits eines darstellt.

Erst jetzt könnte ins Fenster gezeichnet werden, doch sollte man dies auf keinen Fall „von Hand" tun! Denn nach dem Öffnen des Fensters wird automatisch ein Redraw-Ereignis vom Betriebssystem generiert. Dieses Ereignis wird in der Regel dazu benutzt, um eine Redraw-Routine aufzurufen, die den Fensterinhalt oder Teile davon neu zeichnet. Die Redraw-Routine muß **zu jedem Zeitpunkt** wissen, was gerade im Fenster dargestellt wird, denn sie kann jederzeit aufgerufen werden, um einen Teil des Fensters neu zu zeichnen.

Bei einem Textfenster (wie z.B. in unserem Programm) genügt es, sich für jede Zeichenposition ein Byte zu merken; ein objektorientiertes Programm müßte für jedes Objekt eine entsprechechend aufgebaute Struktur verwalten, und bei einem Malprogramm müßte sogar der gesamte Fensterinhalt als Bitmap im Hintergrund gehalten werden.

Hat sich der Fensterinhalt geändert, muß die Information nur im Hintergrund aktualisiert werden; anschließend genügt es, die Redraw-Routine „manuell" aufzurufen. Das kann zum Beispiel der Fall sein, wenn ein Scroll-Balken betätigt wurde, dazu aber später noch mehr.

American Graffiti

Jetzt jedoch zum Redraw selbst. In folgenden Fällen fordert das AES das Programm auf, Teile oder den gesamten Fensterinhalt neu zu zeichnen:

1. Das Fenster wurde gerade geöffnet.
2. Teile des Fensterinhalts waren bisher verdeckt und sind jetzt sichtbar geworden, z.B. weil ein Fenster nach vorne gebracht oder verschoben wurde.
3. Das Fenster wurde vergrößert. Hierbei ist anzumerken, daß **keine** Redraw-Message erzeugt wird, wenn das Fenster verkleinert wurde! Verwaltet man in seinem Fenster beispielsweise eine Anzahl von Icons, die bei einer Größenänderung des Fensters in ihrer Position entsprechend angepaßt werden sollen, muß der Fall der Fensterverkleinerung vom Programmierer also selbst abgeprüft und der Fensterinhalt gegebenenfalls neu gezeichnet werden.

Für jedes Fenster wird eine Rechteckliste verwaltet, die die sichtbaren Teile eines Fensters in Form von Rechtecken beschreibt. Ein weitverbreiteter Irrtum ist, daß eine Rechteckliste nur nach einem Redraw-Ereignis abgefragt werden könne. Tatsächlich kann sie jedoch jederzeit abgefragt werden! Sie dient dazu, beim Zeichnen nur diejenigen Bereiche auch neu zu zeichnen, die gezeichnet werden **dürfen**, weil sie sichtbar sind. Hierzu wird dem VDI ein sogenannter Clipping-Bereich mitgeteilt (ebenfalls ein Rechteck), auf den alle folgenden Ausgaben beschränkt werden, bis der Clipping-Bereich wieder abgeschaltet oder verändert wird. Ausgaben auf Bereiche außerhalb des Clipping-Bereiches zeigen dabei keine Wirkung. So wird vermieden, daß „aus Versehen" andere Fenster zerstört werden. Um nun aber die Rechtecke zu ermitteln, die wirklich neu gezeichnet werden **müssen**, müssen sämtliche Rechtecke in der Rechteckliste mit dem bei der Redraw-Message mitgelieferten Rechteck geschnitten werden. Nur die so berechneten Bereiche müssen letztendlich gezeichnet werden.

An dieser Stelle ist ein weiterer Trick zur Beschleunigung der Textausgabe angebracht, der eine sehr deutliche Geschwin-

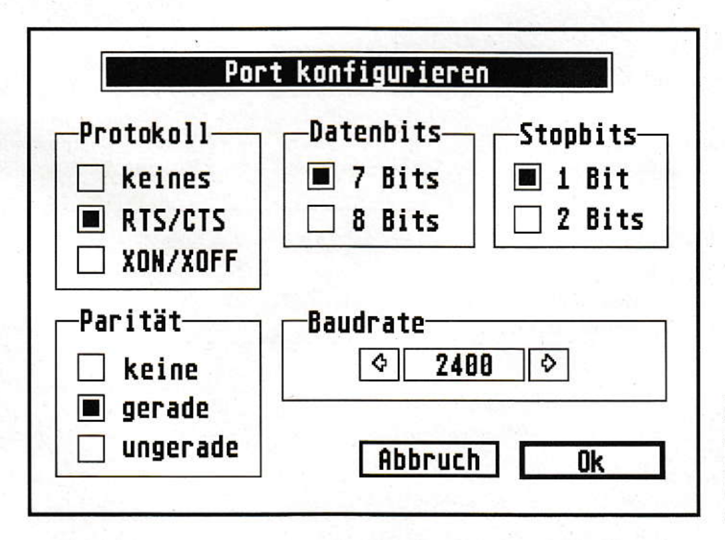


Abb. 1: Mit diesem Dialog können die bis zu vier seriellen Schnittstellen einzeln (und unterschiedlich) konfiguriert werden.

digkeitssteigerung bringt. Die VDI-Funktion *v_gtext()* prüft nämlich bei **jedem** auszugebenden Zeichen ab, ob es im Clipping-Bereich liegt. Paßt jedoch der gesamte String in den Clipping-Bereich, so wird überhaupt nicht geprüft, ob ein Zeichen noch innerhalb des Clipping-Bereiches liegt oder nicht. Die sich dadurch ergebende Zeitersparnis ist enorm!

Um dies ausnutzen zu können, müssen jedoch einige Voraussetzungen gegeben sein: Einerseits muß außer der x-Koordinate auch die Breite des Fensterarbeitsbereiches auf Zeichengrenzen eingestellt werden, und andererseits muß das neu zu zeichnende Fenster in voller Breite sichtbar sein. Damit *v_gtext()* auch erkennt, daß der auszugebende Text ganz in den Clipping-Bereich hineinpaßt, wird dessen Breite einfach noch etwas breiter gemacht. Ist die auszugebende Zeichenkette länger, als das Fenster breit ist, muß sie natürlich an der richtigen Stelle abgeschnitten werden, damit ohne Clipping nicht über den rechten Fensterrand hinausgeschrieben wird.

Maverick & Goose, Iceman & Slider...

Einer der Vorteile, die Fenster bieten, besteht darin, daß es möglich ist, mit seiner Hilfe mehr Daten darzustellen, als eigentlich auf den Bildschirm passen. Um an diese Daten heranzukommen, sind die allgemein bekannten Rollbalken, neudeutsch auch „Slider" genannt, eingeführt worden. Wie man diese in ihrer richtigen Größe an der richtigen Stelle darstellt, ist auf den ersten Blick (und auch auf den zweiten) gar nicht mal so einfach.

Im folgenden wollen wir uns nur auf den vertikalen Slider beziehen, alle Angaben gelten in analoger Weise jedoch auch für den horizontalen.

Das eigentliche Problem besteht nämlich nicht darin, sie anzuzeigen, sondern ihre Größe und Position zu verwalten. Wenden wir uns zuerst der Größe zu. Die Größe wird durch Werte von 1 bis 1000 repräsentiert, wobei 1 die kleinste Slider Größe darstellt; bei einer Slider-Größe von 1000 bedeckt der Slider den gesamten ihm zur Verfügung stehenden Bereich. Die erste Hürde besteht darin, das Verhältnis von gesamter Objektgröße zum dargestellten Teil in Werte von 1 bis 1000 umzurechnen.

Die Formel für diese Umrechnung sieht in C-Syntax jedoch relativ einfach aus:

```
groesse = min (1000L, 1000L * sichtbarer_teil /
        gesamtgroesse);
```

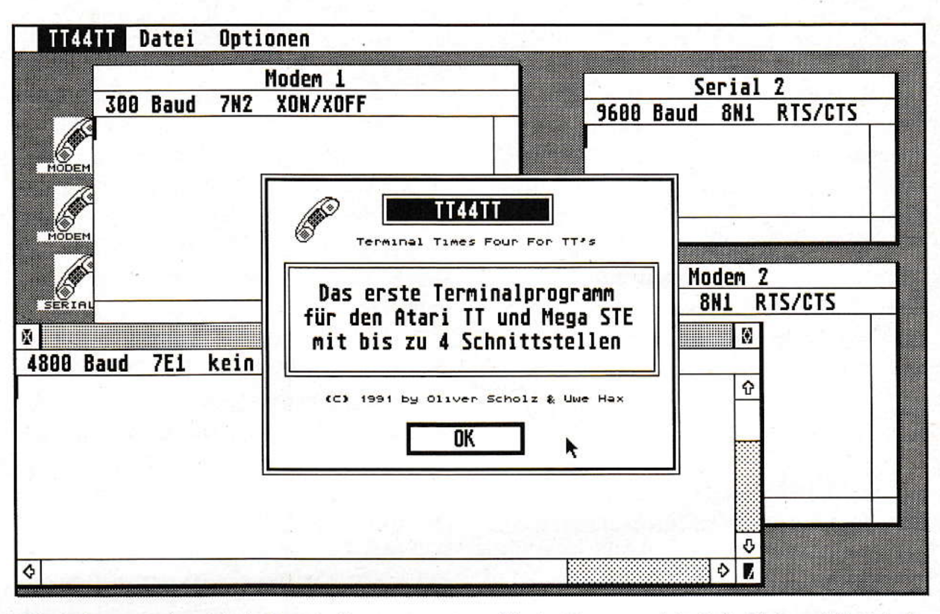


Abb. 2: So präsentiert sich das Terminalprogramm, wenn alle vier Fenster und der Info-Dialog geöffnet sind.

Dabei müssen natürlich die Variablen *sichtbarer_teil* und *gesamtgroesse* die gleichen Einheiten besitzen; würde die Variable *sichtbarer_teil* beispielsweise die Höhe des Fensters in Pixeln und *gesamtgroesse* die Gesamtlänge eines Textes in Zeilen angeben, so ginge die obige Umrechnung garantiert schief.

Warum nun das *min()* in der Formel? Die Antwort ist ganz einfach: Ist *sichtbarer_teil* (d.h. das Fenster) größer als *gesamtgroesse*, würde der rechte Teil der Formel einen Wert größer als 1000 liefern, was jedoch nicht zulässig ist. Ebenso sollte darauf geachtet werden, daß *gesamtgroesse* nicht 0 wird; in diesem Fall sollte *groesse* ebenfalls auf den Wert 1000 gesetzt werden.

Unbedingt sollte darauf geachtet werden, daß mit Langwörtern gerechnet wird, da sonst sehr leicht der Integer-Zahlenbereich verlassen werden kann. Laut Kernighan & Ritchie sollte es genügen, wenn wie oben ein Operand vom Typ „long" ist. Der Compiler sollte dann die übrigen Operanden ebenfalls nach „long" wandeln; darauf kann man sich jedoch nicht bei jedem Compiler unbedingt verlassen. In Zweifelsfällen sollten deshalb alle Operanden mittels Casting in den Typ „long" umgewandelt werden.

Die so berechnete Größe kann dann mittels *wind_set(..., WF_VSLSIZE,...)* (bzw. WF_HSLSIZE bei horizontalem Slider) gesetzt werden.

Wird das Fenster in der Größe verändert, generiert das AES eine WM_SIZED-Message. Daraufhin sollten auch die Slider-Größen neu berechnet und eingestellt werden. Ansonsten ist eine Berechnung der Slider-Größen nur notwendig, wenn das Fenster zum ersten Mal geöffnet wurde oder sich die Gesamtlänge des dargestellten Objektes geändert hat (z.B. wenn in einem Text eine Zeile eingefügt wurde).

Die Position des Sliders wird ebenfalls durch eine Zahl von 1 bis 1000 dargestellt. Die Formel zur Positionsberechnung eines Sliders sieht so aus:

```
position = 1000L * anfang / ausserhalb;
```

Dabei ist *anfang* der Anteil, der vor dem Beginn des Fensters liegt, und *ausserhalb* der Gesamtanteil, der außerhalb des Fensters liegt; er berechnet sich in der Regel aus der Gesamtgröße abzüglich des sichtbaren Teils. Ein Beispiel soll das verdeutlichen: Nehmen wir an, im Fenster soll ein Text dargestellt werden, der 25 Zeilen lang ist, das Fenster kann zur Zeit jedoch nur 9 Zeilen darstellen. Die erste im Fenster sichtbare Zeile sei die Zeile 4 (beginnend mit 0). Daraus ergibt sich, daß *anfang* den Wert 4 (Zeilen 0 bis 3) und *ausserhalb* den Wert 25 - 9 = 16 hat. Somit ergibt sich für *positon* der Wert 250.

Auch die Position des Sliders wird wieder mit Hilfe der Funktion *wind_set (...,WF_VSLIDE,...)* (bzw. WF_HSLIDE für den horizontalen Slider) eingestellt.

Die Position der Slider sollte außer bei Öffnen eines Fensters immer dann neu berechnet und gesetzt werden, wenn sich die Position des Fensterausschnitts relativ zum Gesamtobjekt geändert hat, sei es durch Verändern der Objektgröße oder durch irgendeine Fenstermanipulation von seiten des Benutzers.

Das Anklicken des Sliders selbst kann nur eine Nachricht generieren: WM_-HSLID bzw. WM_VSLID. Dieser liefert die vom Benutzer gewünschte neue Slider-Position im Bereich von 1 bis 1000 zurück. Dieser Wert kann entweder benutzt werden, um den Slider direkt zu setzen, oder erst nach Einflußnahme des Programmierers (z.B. wenn der Slider lediglich in 100er-Schritten verschoben

werden soll). Natürlich muß daraufhin auch der Fensterinhalt neu gezeichnet werden. Das Betriebssystem liefert hier keine Redraw-Message! Meistens kann man jedoch mit der Zahl im Bereich von 1 bis 1000 wenig anfangen; zuerst muß diese Zahl wieder in eine brauchbare Form umgerechnet werden. Dazu benötigen wir die Umkehrfunktion der oben aufgeführten Formel, mit der im Textbeispiel die erste sichtbare Zeile im Fenster berechnet werden kann:

```
anfang =  message_buffer[4] * ausserhalb /
          1000L;
```

Die oben angeführten Berechnungen müssen ebenfalls durchgeführt werden, wenn der schraffierte Bereich außerhalb des Sliders angeklickt wird. In diesem Fall erhält man die Meldung WM_ARROWED mit einem der Untertypen WA_UPPAGE, WA_DNPAGE, WA_LFPAGE oder WA_RTPAGE. Mit anderen Worten: es soll um eine Seite in eine der angegebenen Richtungen gescrollt werden. Die Größe einer Seite wird dabei vom Programmierer festgelegt.

Beim Anklicken der Pfeiltasten (sofern vorhanden) wird wiederum die Meldung WM_ARROWED erzeugt, diesmal aber mit einem der Untertypen WA_UPLINE, WA_DNLINE, WA_LFLINE oder WA_RTLINE erzeugt. Hier sollte jeweils um einen entsprechend geringeren Betrag in die angegebene Richtung gescrollt werden.

Bei all diesen Meldungen sollte einerseits die Slider-Position aktualisiert und andererseits auch darauf geachtet werden, daß der Fensterausschnitt nicht über das Ende (oder den Anfang) des Dokuments hinaus gescrollt wird.

Vorhang zu!

Klickt der Benutzer das Schließfeld eines Fensters an, so erzeugt das AES eine WM_CLOSED-Meldung. Das Programm sollte dann das Fenster mit *wind_close()* schließen. Ist das geschehen, befindet sich die Fensterstruktur jedoch immer noch im Speicher und kann mit *wind_open()* jederzeit wieder geöffnet werden. Erst wenn *wind_delete()* aufgerufen wird (unbedingt **nach** Aufruf von *wind_close()*), werden die entsprechenden internen Datenstrukturen des AES freigegeben und sind für andere Fenster verfügbar. Ab TOS 1.04 ist zusätzlich die Funktion *wind_new()* verfügbar, die alle Fenster schließt und löscht.

Somit haben wir jetzt den vollständigen „Lebenszyklus" eines Fensters durchlaufen. Die Verwaltung von mehr als einem Fenster ist jedoch auch nicht viel schwieriger. Bei allen Nachrichten, die mit Fenstern zu tun haben, liefert das AES das Handle des angesprochenen Fensters mit. Der Programmierer möchte in der Regel mit Fensternummern arbeiten und nicht mit Handles, so daß es sich als praktisch erwiesen hat, ein Array anzulegen. In dieses wird für jedes Fenster das entsprechende Handle eingetragen, wodurch es möglich wird, schnell und einfach vom Handle auf Fensternummer umzurechnen und umgekehrt.

Da Fenster-Handles normalerweise jedoch nur Werte von 0 bis 7 annehmen können, gibt es einige Programmierer, die auf die Umrechnung von Handles in Fensternummern verzichten und den entsprechenden Array-Eintrag direkt ansprechen. Das sollte man jedoch auf gar keinen Fall tun! Erstens ist nirgends dokumentiert, daß man nur Werte von 0 bis 7 als Handles geliefert bekommt (das kann sich in zukünftigen TOS-Versionen schnell ändern), und zweitens gibt es bereits Programme, die GEM wesentlich mehr Fenster zur Verfügung stellen (siehe [1]) und demzufolge auch größere Handles liefern.

Es bietet sich an, in dem genannten Array gleich weitere fensterspezifische Informationen abzulegen, die für die Fensterverwaltung benötigt werden. In unserem Programm ist das ein Array vom selbstdefinierten Typ WINDOW, der alle benötigten Werte enthält. Bei einem Mausklick in ein Fenster wird das Handle des betroffenen Fensters zwar nicht angegeben, jedoch läßt es sich leicht mit der Funktion *wind_find()* ermitteln.

Kommen wir jetzt noch zur letzten, bisher nicht erwähnten Fensterfunktion: *wind_update()*. Sie wird immer dann benötigt, wenn in ein Fenster (bzw. auf den Bildschirm) gezeichnet werden soll. Wird nämlich *wind_update(BEG_UPDATE)* aufgerufen, ist es nicht mehr möglich, daß andere Programme inklusive des AES den Bildschirm verändern und möglicherweise eigene Zeichenoperationen stören; das betrifft insbesondere Drop-Down-Menüs. Ist das Zeichnen ins Fenster beendet, muß anschließend ein *wind_update(END_UPDATE)*-Aufruf erfolgen. Ansonsten bleibt die Bildschirmausgabe für andere Programme auf ewig gesperrt.

Es sollte auch nicht vergessen werden, vor allen Bildschirmausgaben die Maus abzuschalten (und hinterher wieder einzuschalten). Manchmal kann es notwendig sein, daß ein Programm **alle** Mausklicks erhalten soll, also auch die, die auf Randelemente eines Fensters oder auf die Menüleiste erfolgen. Solche Programmteile müssen dann mit *wind_update(BEG_MCTRL)* und *wind_update(END_MCTRL)* geklammert werden. Das bewirkt also nichts anderes, als daß für die Dauer dieses Programmteils der Bildschirm-Manager abgeschaltet wird. Diese Funktion benötigt man beispielsweise, wenn man einen eigenen Dialog-Handler schreiben möchte.

Des Desktops neue Kleider

Jetzt wissen wir also, wie wir Fenster zu öffnen und zu verwalten haben. Leider liegen sie aber noch auf dem bekannten grauen (oder grünen) Hintergrund. Wie aus vielen Programmen (u.a. Ataris Desktop) bekannt ist, ist es zusätzlich möglich, auf dem Hintergrund Icons anzulegen. Dazu stellt das AES mit der Funktion *wind_set(...,WF_NEWDESK,...)* die Möglichkeit zur Verfügung, einen eigenen Objektbaum als Desktop anzumelden. Dies hat gegenüber einem selbstverwalteten Hintergrund den Vorteil, daß sämtliche Redraws vom AES übernommen werden. Dazu muß im Resource Construction Set ein normaler Dialog angelegt werden, der später als Desktop dargestellt werden kann. Da bei der Erstellung des Desktops keine Annahmen über die später verwendete Bildschirmgröße gemacht werden können, sollten Breite und Höhe des Desktops nach Programmstart ermittelt und in die *ob_width*- und *ob_height*-Komponenten des Wurzelobjektes des neuen Desktops eingetragen werden. Das Programm muß anschließend nur noch den Objektbaum einmal mit *objc_draw()* zeichnen. Da ein solcher Objektbaum ein Standard-Objektbaum ist, kann er nicht nur Icons beinhalten, sondern auch beliebige andere Objekte, wie z.B. Buttons oder benutzerdefinierte Objekte. In unserem Programm haben wir jedoch der Einfachheit halber nur Icons verwendet.

Ist bei Start eines Programmes nicht bekannt, wieviel Icons insgesamt verwendet werden sollen, muß der Objektbaum die maximal mögliche Zahl Icons enthalten; nicht benötigte Icons können dann einfach durch Setzen des HIDETREE-Flags von der Oberfläche entfernt werden. In unserem Terminalprogramm werden so je nach Rechnertyp nur die Icons angezeigt, für die auch eine entsprechende Schnittstelle vorhanden ist. Beim Anklicken eines Icons wird dieses im Gegensatz zu Dialogboxen **nicht** automatisch selektiert; das muß vom Programm übernommen werden. Dazu wird bei *evnt_multi()* u.a. auf Mausklicks gewartet. Ist ein Mausereignis eingetreten, muß mittels der *wind_find()*-Funktion das Fenster ermittelt werden, das sich unter den Mauskoordinaten befindet. Handelt es sich dabei um das Desktop, wird das Fenster-Handle 0

zurückgegeben. Nun wissen wir also, daß der Mausklick auf das Desktop erfolgt ist. Den benötigten Objektindex ermittelt jetzt die Funktion *objc_find()*. Soll das Objekt selektiert dargestellt werden, genügt es nicht, das SELECTED-Flag zu setzen, das Objekt muß auch noch selbst gezeichnet werden. Das nimmt uns das AES leider nicht ab, da es natürlich keine Kenntnis darüber hat, daß sich das entsprechende Flag geändert hat.

Hier darf im Normalfall allerdings nicht - wie von Dialogboxen gewohnt - die Funktion *objc_change()* oder *objc_draw()* benutzt werden, um das Objekt selektiert darzustellen. Denn es ist ohne weiteres möglich, daß ein Icon teilweise von einem Fenster verdeckt ist. Ein Neuzeichnen des gesamten Objektes würde in diesem Fall einen Teil des Fensters zerstören. Es darf also nur der Teil des Icons neu gezeichnet werden, der nicht von Fenstern verdeckt wird. Da das Desktop jedoch ein Fenster wie (fast) jedes andere ist, verfügt es auch über eine Rechteckliste, die wie üblich abgefragt werden kann. Jetzt muß jedes Rechteck in der Liste mit dem Objektrechteck geschnitten werden. Existiert ein Schnittbereich, wird er als Clipping-Bereich für *objc_draw()* benutzt und der entsprechende Teil des Icons neu (selektiert) gezeichnet. Analog ist beim Deselektieren vorzugehen.

Soll ein Icon verschoben werden (was in unserem Programm nicht möglich ist), genügt es, die x- und y-Koordinate des Icons entsprechend zu verändern und anschließend den Objektbaum an der neuen und alten Position mit oben beschriebener Methode neu zu zeichnen. Es ist dabei völlig irrelevant, ob ein Icon ganz oder teilweise von einem Fenster verdeckt ist, da die obige Methode das berücksichtigt.

Behind the Scenes

Jetzt kommen wir endlich zur Programmbeschreibung. Wer fleißig die Listings der letzten Teile abgetippt hat, wird schon wissen, daß das Programm aus vier Modulen besteht. Der Rest der Welt weiß es eben jetzt. Die Aufgabenverteilung ist dabei wie folgt:

- TT44TT.C enthält die Initialisierung, die Ereignisverwaltung und den Dispatcher.
- OUTPUT.C erledigt die Ausgabe der Zeichen in die Fenster.
- WINDOW.C kümmert sich um die Fensterverwaltung.
- CONFIG.C schließlich stellt die Dialogbox zur Einstellung der Schnittstellenparameter auf dem Bildschirm dar und liest die neue Einstellung gegebenenfalls ein.

Die globalen Variablen befinden sich in der Header-Datei VARIABLE.H. Dazu sind sie als GLOBAL deklariert, was in allen Modulen außer dem Hauptmodul als „extern" definiert ist. Das hat den Vorteil, daß die globalen Variablen nicht über alle Module verteilt sind, sondern zentral in einer Datei existieren. Dieses Verfahren stammt aus [2]. Sämtliche Prototypen befinden sich in der Header-Datei PROTO.H. In TT44TT.H liegen die vom RCS erzeugten Objektindizes. Die restlichen (allgemeinen) Definitionen befinden sich im Header TERMDEFS.H.

Beginnen wir mit dem Modul TT44TT.C. Was soll eigentlich dieser komische Name? Ganz einfach, er ist die Abkürzung für „Terminal Times Four For TT's". Der erste interessante Punkt ist die Bestimmung der Anzahl der seriellen Schnittstellen in der Funktion *main()*. Abhängig von der TOS-Version wird dazu die Funktion *Bconmap()* (siehe Teil 1 dieser Serie) benutzt, ansonsten wird angenommen, daß nur die beim ST übliche eine Schnittstelle vorhanden ist.

Als nächstes wird mittels *Giaccess()* geprüft, ob der LAN-Port aktiv ist. Ist das der Fall, steht eine Schnittstelle weniger zur Verfügung. Übrigens läßt sich der LAN-Port durch *Offgibit(0x7f)* ein- und durch *Ongibit(0x80)* wieder ausschalten.

Des weiteren werden in *main()* ein eigenes Desktop installiert und die Fenster initialisiert. Darauf wollen wir hier jedoch nicht weiter eingehen, da wir die Vorgehensweise oben ausführlich beschrieben haben.

Nach den Initialisierungen wird dann die Funktion *events()* aufgerufen, die die Hauptschleife und das unvermeidliche *evnt_multi()* enthält. Und dabei kommen wir auch schon zum eigentlichen Clou des Programmes.

Normalerweise kommunizieren Programme nämlich nur über das aktive (d.h. oberste) Fenster. Da das AES für jedes Fenster zu jedem Zeitpunkt eine Rechteckliste verwaltet (wie oben beschrieben), ist es jedoch ohne weiteres möglich, auch in inaktive Fenster auszugeben. Und genau das wird hier gemacht. Das Programm muß hierzu merken, wann in welches Fenster welche Daten ausgegeben werden müssen und diese an der richtigen Position auch ausgeben.

Die Eingabe kann dagegen nicht in alle Fenster gleichzeitig erfolgen; daher werden die Zeichen, die von der Tastatur kommen, immer an die zum aktiven Fenster gehörende Schnittstelle gesendet. Wer sich wundert, warum die eingetippten Zeichen nicht im Fenster erscheinen, dem sei gesagt, daß es sich nicht um eine Textverarbeitung, sondern um ein Terminalprogramm ohne Echo-Funktion handelt. Mit anderen Worten: alle eingegegebenen Zeichen müssen von der Gegenseite (z.B. Mailbox) zurückgesendet werden.

Special Effects

Jetzt zurück zur Ausgabe: Alle 100 Millisekunden (ausgelöst durch *evnt_multi()*) werden alle Schnittstellen überprüft, ob Zeichen zur Ausgabe anliegen. Ist das der Fall, wird die Funktion *do_output()* aufgerufen, um den Puffer zu entleeren und ins Fenster auszugeben. Es wird also quasi Multitasking ausgeführt, auch wenn es sich dabei nur um das eingeschränkte Multitasking handelt, das GEM zur Verfügung stellt. Die Routine *do_output()* befindet sich (logischerweise) im Modul OUTPUT.C. Um jederzeit in der Lage zu sein, den Fensterinhalt neu zu zeichnen, gehört zu jedem Fenster ein „virtuelles Terminal". Dabei handelt es sich um ein zweidimensionales Feld vom Typ „char", das ein 80 x 25 Zeichen großes Terminal darstellt. Alle Ausgaben geschehen zunächst in dieses virtuelle Terminal, so daß zum Beispiel bei einem Linefeed hier der gesamte Inhalt um eine Zeile nach oben kopiert werden muß. Das ist notwendig, da das virtuelle Terminal die Grundlage für alle Fensterausgaben darstellt.

Leider war es wegen der Länge des Listings nicht möglich, einen bestimmten Terminaltyp zu realisieren. Das Programm ist als Ausgangspunkt für ein eigenes Terminalprogramm zu sehen, in dem die gewünschte Terminalemulation eingebaut werden kann. Dazu ist bereits die Funktion *handle_escape()* vorgesehen, die allerdings bisher noch ein karges Dasein fristet.
Des weiteren wären folgende Erweiterungen denkbar:

- ein History-Puffer, der auch über die letzten 25 Zeilen hinausgehende Ausgaben puffert
- eine Up/Download-Möglichkeit, evtl. mit verschiedenen Protokollen
- eine Funktionstastenbelegung mit Makros
- ...

Wer das Programm erweitern möchte, wird früher oder später nicht darum herumkommen, neue Menüpunkte und Dialoge einzubauen. Wir haben uns deshalb dazu entschlossen, die RSC-Datei mit auf die Diskette zum Heft zu packen, da größere Modifikationen in der RSH-Datei fast unmöglich sind. Außerdem haben wir das Programm im Monochrommodus entwikkelt, weshalb die Dialogboxen in einer anderen Auflösung etwas „durcheinandergewürfelt" aussehen. Dies war ein weiterer Anlaß, die Resource-Datei für eigene Änderungen beizufügen.

Erwähnenswert ist an dieser Stelle noch das vertikale Scrolling in den Fenstern. Anstatt jedes Mal den gesamten Fensterinhalt mit *v_gtext()* neu auszugeben (was viele Programme tun), kopieren wir stattdessen mit Hilfe von *vro_cpyfm()* soviel wie möglich und geben nur unbedingt notwendige Zeichen mit *v_gtext()* aus. Im Gegensatz dazu haben wir beim horizontalen Scrolling nicht solchen Aufwand getrieben, da dieser Fall relativ selten vorkommt und nur von Hand im aktiven Fenster ausgelöst werden kann. Dort ist jedoch die weiter oben beschriebene schnelle Textausgabe mit Abschalten des Clippings möglich.

Im Modul WINDOW.C befindet sich die Behandlung der fensterbezogenen Ereignisse. Wer den ersten Teil dieses Artikels aufmerksam gelesen hat, sollte keine Probleme haben, die Funktionsweise der einzelnen Funktionen zu durchschauen.

Das Modul CONFIG.C enthält schließlich die komplette Dialogbehandlung zur Konfiguration der Schnittstellen und benutzt nur hinreichend oft beschriebene Routinen zur Dialogbehandlung, auf die wir hier nicht zum x-ten Male eingehen wollen.

Make-Up

Nachdem wir die Funktionsweise des Programms in groben Zügen besprochen haben, kommen wir noch zu dessen Bedienung. Nach Programmstart erscheint auf dem Desktop für jede Schnittstelle ein Icon. Wird ein Icon einfach angeklickt, ist es selektiert und im Menü kann unter „Optionen/Parameter..." die Schnittstelle konfiguriert werden. Ist kein Icon selektiert, bezieht sich der Parameter-Dialog (siehe Abbildung 1) auf das jeweils aktive Fenster (sofern vorhanden).

Bei einem Doppelklick auf ein Icon wird das zur Schnittstelle gehörige Fenster geöffnet und das entsprechende virtuelle Terminal gelöscht. Alternativ läßt sich ein Fenster wie im Desktop auch dadurch öffnen, daß ein Icon selektiert und anschließend der Menüpunkt „Datei/Öffnen" angewählt wird. Ist das angesprochene Fenster bereits geöffnet, wird es bei der Anwahl des Icons lediglich in den Vordergrund gebracht. Ein Beispiel mit allen vier geöffneten Fenstern und dem unvermeidlichen Info-Dialog zeigt Abbildung 2.

Damit nach einem Programmstart die gewohnte Arbeitsumgebung wiederhergestellt wird, können mit den Menüpunkten „Optionen/INF-Datei laden..." bzw. „Optionen/INF-Datei sichern..." alle augenblicklich eingestellten Parameter geladen bzw. gespeichert werden. Betroffen sind davon die Einstellungen im Parameter-Dialog und die Fensterpositionen. Nach einem Programmstart wird die INF-Datei mit dem Namen „TT44TT.INF" (sofern vorhanden) automatisch eingeladen.

The Making of...

Mit Hilfe der abgedruckten Projektdatei läßt sich aus den Quelltexten unter Turbo-C problemlos das lauffähige Programm erzeugen. Voraussetzung ist natürlich, daß Sie alle Listings dieser Serie fehlerfrei abgetippt haben oder sich im Besitz der Diskette zum Heft befinden.

Damit wollen wir nun diesen Artikel (und damit vorläufig auch diese Serie) beenden. Wir hoffen, daß wir damit vielen Programmierern weitergeholfen und die (neuen) Schnittstellen des Mega STE/TT etwas transparenter gemacht haben.

Oliver Scholz & Uwe Hax

Literatur:

[1] WINX - Viele Fenster zum Hof, ST-Computer 9/91, S. 195

[2] Dieter & Jürgen Geiß: Vom Anfänger zum GEM-Profi, Hüthig Verlag

Korrektur

Leider sind uns im zweiten Teil unserer Schnittstellenserie Fehler im Listing OUTPUT.C unterlaufen. Die Zeile 157 ist folgendermaßen zu ändern:

```
CHAR nonprintable[4];
```

Nach Zeile 165 ist folgende Zeile einzufügen:

```
nonprintable[3]=EOS;
```

```
/*
 * TT44TT.PRJ
 * Projektdatei für Turbo-C
 * Copyright (c) 1991 by MAXON Computer
 * Autoren: Oliver Scholz & Uwe Hax
 */

TT44TT.PRG          ;Name des erzeugten Programms

=

TCSTART.O           ;Startup-Code

TT44TT.C            ;die vier Module
WINDOW.C
CONFIG.C
OUTPUT.C

TCSTDLIB.LIB        ;Standard-Bibliothek
TCTOSLIB.LIB        ;TOS-Bibliothek
TCGEMLIB.LIB        ;AES- und VDI-Bibliothek
```

```
/*
 * PROTO.H
 * Prototyp-Header für TT44TT
 * Copyright (c) 1991 by MAXON Computer
 * Autoren: Oliver Scholz & Uwe Hax
 */

VOID clipping(GRECT *rect,WORD mode);
VOID do_output(WORD wind_index);
VOID init_terminal(WORD wind_index);
VOID handle_escape(WORD wind_index,CHAR c);
VOID insert_char(WORD wind_index, CHAR c);
VOID print_char(WORD wind_index, CHAR c);
VOID term_lf(WORD wind_index);
VOID scroll(WORD wind_index, WORD direction);
VOID pos_slider(WORD wind_index, WORD vh_flag);
WORD rc_intersect(GRECT *r1,GRECT *r2);
VOID cursor(WORD wind_index, WORD flag);
VOID init_dial(VOID);
VOID into_dial(CONF_RS *port, OLDSET *indices);
VOID read_port(WORD index);
VOID get_baud_string(WORD rate_index, CHAR *buf);
VOID write_port(WORD index);
VOID main(VOID);
VOID open_vwork(WORD phys_handle);
VOID get_addresses(VOID);
WORD open_window(WORD i);
VOID wind_snap(WORD *x,WORD *y,WORD *w,WORD *h);
VOID events(VOID);
WORD mn_selected(WORD *mesg_buff);
VOID wm_topped(WORD whandle);
VOID wm_redraw(WORD whandle,
               WORD x,WORD y,WORD w,WORD h);
VOID wm_moved(WORD *mesg_buff);
VOID wm_fulled(WORD *mesg_buff);
VOID wm_arrowed(WORD *mesg_buff);
VOID wm_vslid(WORD *mesg_buff);
VOID wm_hslid(WORD *mesg_buff);
VOID wm_closed(WORD whandle);
VOID wm_sized(WORD *mesg_buff);
VOID wind_max(WORD *x,WORD *y,WORD *w,WORD *h);
VOID clipping(GRECT *rect,WORD mode);
WORD rc_intersect(GRECT *r1,GRECT *r2);
VOID show_info(VOID);
VOID ienable(WORD flag);
VOID draw_icon(WORD iconidx);
WORD fileselect (CHAR *inpath, CHAR *insel,        →
```

```
                WORD *exbutton, CHAR *label);
VOID loadinf(CHAR *pfad,WORD init);
VOID saveinf(CHAR *pfad);
VOID get_tos_version(VOID);
VOID wind_info(WORD device);
VOID scroll(WORD wind_index,WORD direction);
VOID get_baud_string(WORD rate_index,
                     CHAR *buf);
VOID size_slider(WORD wind_handle);
VOID init_ports(CONF_RS port[4]);
WORD conf_port(CONF_RS *port);
VOID pos_slider(WORD wind_index, WORD vh_flag);
VOID do_output(WORD wind_index);
VOID init_terminal(WORD wind_index);
VOID read_port(WORD device);
VOID write_port(WORD device);
WORD get_index(WORD whandle);
VOID adjust(WORD whandle);
VOID update_cursor(WORD wind_index);
VOID update_pos(WORD wind_index);
LONG _bconmap(WORD devno);
OBJECT *get_traddr(WORD tree_index);
```

```
/*
 * TERMDEFS.H
 * Definitionen für TT44TT
 * Copyright (c) 1991 by MAXON Computer
 * Autoren: Oliver Scholz & Uwe Hax
 */

/* Alarmboxen */

#define APP_NOT_STARTED "[3][Applikation konnte
           nicht|initialisiert werden!][ Ok ]"
#define NO_WINDOW       "[1][Es konnte kein
     weiteres|Fenster mehr geöffnet werden!][ Ok ]"
#define INF_WRERR       "[1][Kann INF-Datei
                  nicht schreiben][OK]"
#define INF_RDERR       "[1][Kann INF-Datei
                  nicht lesen!][OK]"

/* allgemeine Definitionen */

#define TRUE            1
#define FALSE           0
#define DESKTOP         0
#define EOS             0
#define HORIZONTAL      1
#define VERTICAL        0
#define CURSOR_OFF      0
#define CURSOR_ON       1
#define SCROLL_UP       0
#define SCROLL_DOWN     1

#define CHAR            char
#define _sysbase        0x4F2L

/* Terminal-Definitionen */

#define TERM_WIDTH      80
#define TERM_HEIGHT     25
#define WAITING         0

/* ASCII Codes */

#define BELL            7
#define BACKSPACE       8
#define TAB             9
#define LF              10
#define CR              13
#define ESCAPE          27
#define DELETE          127

/* Handshakearten für rsconf() */

#define P_NONE  0
#define P_XON   1
#define P_RTS   2

/* Default UCR: 8N1 */

#define DEFUCR  0x88

/* verschiedene Makros */

#define show_mouse()  graf_mouse(M_ON,0L)
#define hide_mouse()  graf_mouse(M_OFF,0L)
#define min(a,b)      ((a)<(b) ? (a) : (b))
#define max(a,b)      ((a)>(b) ? (a) : (b))

/* Typdefinitionen */

typedef struct {  /* Einstellung eines Ports */
    int baudrate;
    int flowctrl;
    int ucr;
} CONF_RS;

typedef struct {  /* alte Porteinstellung */
    int iflow;
    int idata;
    int istop;
    int ipar;
} OLDSET;

typedef struct { /* Virtuelles Terminal */
        CHAR screen[TERM_HEIGHT][TERM_WIDTH+1];
        WORD x,y;
        WORD escape;
        BYTE auxiliary;
        WORD tmp_x,tmp_y;
} TERMINAL;

typedef struct { /* Fensterinformationen */
    WORD handle;
    CHAR title[TERM_WIDTH];
    CHAR info[TERM_WIDTH];
    WORD x,y,w,h;
    WORD fulled;
    WORD x_corner,y_corner;
} WINDOW;
```

```
/*
 * TT44TT.H
 * Resource-Header für TT44TT
 * Copyright (c) 1991 by MAXON Computer
 * Autoren: Oliver Scholz & Uwe Hax
 */

#define PORTS 0          /* TREE */
#define ABOUTBOX 1       /* TREE */
#define NEWDESK 2        /* TREE */
#define MENU 3           /* TREE */
#define RTSCTS 5         /* OBJECT in TREE #0 */
#define NOPROT 6         /* OBJECT in TREE #0 */
#define XONXOFF 7        /* OBJECT in TREE #0 */
#define PARO 11          /* OBJECT in TREE #0 */
#define PARN 12          /* OBJECT in TREE #0 */
#define PARE 13          /* OBJECT in TREE #0 */
#define BAUDRATE 19      /* OBJECT in TREE #0 */
#define BAUDUP 20        /* OBJECT in TREE #0 */
#define BAUDDWN 21       /* OBJECT in TREE #0 */
#define BITS8 24         /* OBJECT in TREE #0 */
#define BITS7 26         /* OBJECT in TREE #0 */
#define PORTABRT 29      /* OBJECT in TREE #0 */
#define STOP2 33         /* OBJECT in TREE #0 */
#define STOP1 35         /* OBJECT in TREE #0 */
#define PORTOK 37        /* OBJECT in TREE #0 */
#define ABOUTOK 9        /* OBJECT in TREE #1 */
#define MFP2 1           /* OBJECT in TREE #2 */
#define SCC1 2           /* OBJECT in TREE #2 */
#define SCC2 3           /* OBJECT in TREE #2 */
#define MFP1 4           /* OBJECT in TREE #2 */
#define ABOUT 8          /* OBJECT in TREE #3 */
#define OPEN 17          /* OBJECT in TREE #3 */
#define CLOSE 18         /* OBJECT in TREE #3 */
#define QUIT 20          /* OBJECT in TREE #3 */
#define PORT 22          /* OBJECT in TREE #3 */
#define ZOOM 24          /* OBJECT in TREE #3 */
#define LOADINF 26       /* OBJECT in TREE #3 */
#define SAVEINF 27       /* OBJECT in TREE #3 */
```

```
/*
 * VARIABLE.H
 * Globale Variablen für TT44TT
 * Copyright (c) 1991 by MAXON Computer
 * Autoren: Oliver Scholz & Uwe Hax
```

→

```
*/

GLOBAL WORD gl_apid;
GLOBAL WORD distances[5],effects[3];
GLOBAL WORD num_aux,aux_offset;
GLOBAL WORD curr_icon=-1;
GLOBAL WORD curr_device=-1;
GLOBAL WORD top_window=-1;
GLOBAL WORD vdi_handle;
GLOBAL WORD dummy;
GLOBAL WORD zoomflag = TRUE;
GLOBAL WORD wchar,hchar,wbox,hbox;
GLOBAL WORD tos_version;
GLOBAL CHAR inf_path[128];
GLOBAL CHAR inf_name[14];
GLOBAL OBJECT *menu,*newdesk,*port_dial,
        *info_box;
GLOBAL CONF_RS port[4];
GLOBAL TERMINAL terminal[4];
GLOBAL WORD elements=NAME | CLOSER | FULLER |
        MOVER | INFO | SIZER | UPARROW | DNARROW
        | VSLIDE | LFARROW | RTARROW | HSLIDE;
GLOBAL WORD iconlist[4]=
        { MFP1, SCC2, MFP2, SCC1 };
GLOBAL WINDOW window[4]=
{ -1," Modem 1 ","",8,32,304,192,   FALSE,0,0,
  -1," Modem 2 ","",320,32,304,192, FALSE,0,0,
  -1," Serial 1 ","",8,224,304,192,  FALSE,0,0,
  -1," Serial 2 ","",320,224,304,192,FALSE,0,0,
};
```

```
/*
 * TT44TT.RSH
 * Konvertierte Resourcedatei für TT44TT
 * Copyright (c) 1991 by MAXON Computer
 * Autoren: Oliver Scholz & Uwe Hax
 */

WORD IMAG0[] = {
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF,
   0xFFFF, 0xFFFF, 0xFFFF, 0xFFFF
};

WORD IMAG1[] = {
   0x0000, 0x03E0, 0x0000, 0x0C7C,
   0x0000, 0x1186, 0x0000, 0x6203,
   0x0000, 0x8401, 0x0001, 0x0841,
   0x0002, 0x10A1, 0x0004, 0x1152,
   0x0008, 0xF0A2, 0x0011, 0xD844,
   0x0023, 0xCC08, 0x0046, 0xA610,
   0x008F, 0x73E0, 0x011A, 0xAB00,
   0x013D, 0xDC00, 0x022A, 0xA800,
   0x0477, 0x7000, 0x04AA, 0xA000,
   0x091D, 0xC000, 0x110A, 0x8000,
   0x1207, 0x0000, 0x23E2, 0x0000,
   0x2412, 0x0000, 0x490C, 0x0000,
   0x5284, 0x0000, 0x6544, 0x0000,
   0x4AA4, 0x0000, 0x4548, 0x0000,
   0x2290, 0x0000, 0x2120, 0x0000,
   0x1040, 0x0000, 0x0F80, 0x0000
};

ICONBLK rs_iconblk[] = {
   IMAG0, IMAG1, "", 4096,6,23,
   0,0,32,32,16,32,0,7,
   IMAG0, IMAG1, "SERIAL 1", 4096,6,23,
   12,0,32,32, 0,32,57,7,
   IMAG0, IMAG1, "SERIAL 2", 4096,6,23,
   12,0,32,32, 0,32,57,7,
   IMAG0, IMAG1, "MODEM 2", 4096,6,23,
   12,0,32,32,0,32,58,7,
   IMAG0, IMAG1, "MODEM 1", 4096,6,23,
   12,0,32,32,0,32,58,7
};

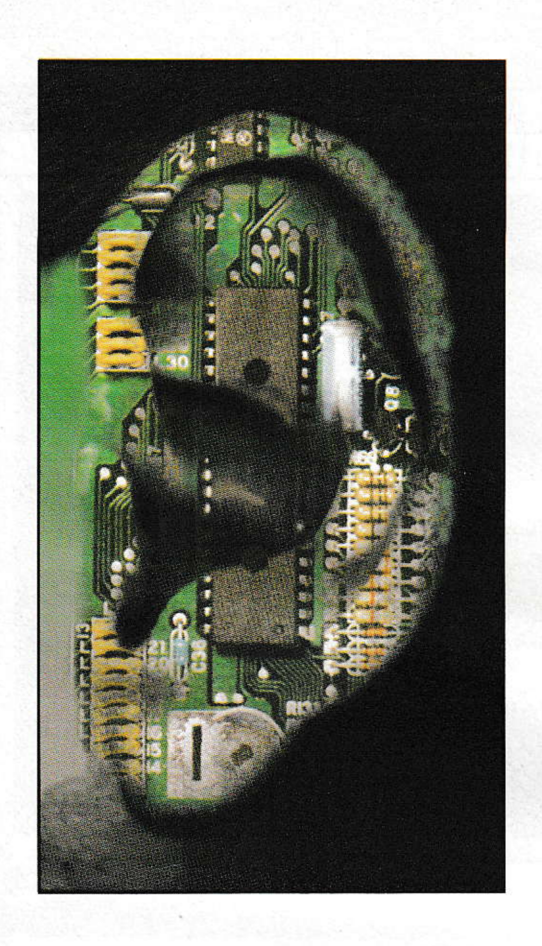
TEDINFO rs_tedinfo[] = {
   "Protokoll", "", "", 3, 6, 0, 0x1180,
   0x0, -1, 10,1,
   "Parität", "", "", 3, 6, 0, 0x1180,
   0x0, -1, 8,1,
   "Port konfigurieren", "", "", 3, 6, 2, 0x1071,
   0x0, -1, 19,1,
   "19200", "", "", 3, 6, 2, 0x1180,
   0x0, -1, 6,1,
   "Baudrate", "", "", 3, 6, 0, 0x1180,
   0x0, -1, 9,1,
   "Datenbits", "", "", 3, 6, 0, 0x1180,
   0x0, -1, 10,1,
   "TEXT", "", "", 3, 6, 0, 0x1180,
   0x0, -1, 5,1,
   "2 Bits", "", "", 3, 6, 0, 0x1180,
   0x0, -1, 7,1,
   "1 Bit", "", "", 3, 6, 0, 0x1180,
   0x0, -1, 6,1,
   "Stopbits", "", "", 3, 6, 0, 0x1180,
   0x0, -1, 9,1,
   "TT44TT", "", "", 3, 6, 2, 0x1071,
   0x0, 255, 7,1,
   "Terminal Times Four For TT's", "", "",
   5, 6, 2, 0x1180, 0x0, 255, 29,1,
   "(C) 1991 by Oliver Scholz & Uwe Hax", "", "",
   5, 6, 2, 0x1180, 0x0, 255, 36,1
};

OBJECT rs_object[] = {
  -1, 1, 37, G_BOX, NONE, OUTLINED,
  0x21100L, 0,0, 1325,2064,
  8, 2, 7, G_BOX, NONE, NORMAL,
  0xFF1100L, 2,2563, 1549,773,
  3, -1, -1, G_STRING, NONE, NORMAL,
  (LONG)"keines", 1028,3840, 6,1,
  4, -1, -1, G_STRING, NONE, NORMAL,
  (LONG)"RTS/CTS", 1028,1538, 7,1,
  5, -1, -1, G_STRING, NONE, NORMAL,
  (LONG)"XON/XOFF", 772,3075, 8,1,
  6, -1, -1, G_BOX, 0x51, OUTLINED,
  0xFF1170L, 1537,2306, 257,2304,
  7, -1, -1, G_BOX, 0x51, OUTLINED,
  0xFF1170L, 1537,769, 257,2304,
  1, -1, -1, G_BOX, 0x51, OUTLINED,
  0xFF1170L, 1537,3843, 257,2304,
  15, 9, 14, G_BOX, NONE, NORMAL,
  0xFF1100L, 2,3849, 1549,773,
  10, -1, -1, G_STRING, NONE, NORMAL,
  (LONG)"keine", 1284,257, 6,1,
  11, -1, -1, G_STRING, NONE, NORMAL,
  (LONG)"ungerade", 772,2819, 8,1,
  12, -1, -1, G_BOX, 0x51, OUTLINED,
  0xFF1170L, 1537,4, 257,2304,
  13, -1, -1, G_BOX, 0x51, OUTLINED,
  0xFF1170L, 1537,1025, 257,2304,
  14, -1, -1, G_BOX, 0x51, OUTLINED,
  0xFF1170L, 1537,2562, 257,2304,
  8, -1, -1, G_STRING, NONE, NORMAL,
  (LONG)"gerade", 772,1538, 6,1,
  16, -1, -1, G_TEXT, NONE, NORMAL,
  (LONG)&rs_tedinfo[0], 1027,259, 9,1,
  17, -1, -1, G_TEXT, NONE, NORMAL,
  (LONG)&rs_tedinfo[1], 1027,1545, 7,1,
  18, -1, -1, G_BOXTEXT, NONE, OUTLINED,
  (LONG)&rs_tedinfo[2], 773,1, 34,1,
  22, 19, 21, G_BOX, NONE, NORMAL,
  0xFF1100L, 785,3849, 1817,2562,
  20, -1, -1, G_BOXTEXT, NONE, NORMAL,
  (LONG)&rs_tedinfo[3], 520,3840, 8,1,
  21, -1, -1, G_BOXCHAR, TOUCHEXIT, NORMAL,
  0x3FF1100L, 1552,3840, 1026,1,
  18, -1, -1, G_BOXCHAR, TOUCHEXIT, NORMAL,
  0x4FF1100L, 517,3840, 1026,1,
  27, 23, 26, G_BOX, NONE, CROSSED,
  0xFF1101L, 1041,2563, 1548,3075,
  24, -1, -1, G_STRING, NONE, NORMAL,
  (LONG)"8 Bits", 1028,1538, 6,1,
  25, -1, -1, G_BOX, 0x51, OUTLINED,
  0xFF1170L, 1793,2306, 257,2304,
  26, -1, -1, G_STRING, NONE, NORMAL,
  (LONG)"7 Bits", 1028,3840, 6,1,
  22, -1, -1, G_BOX, 0x51, OUTLINED,
```

→

```
0xFF1170L, 1793,513, 257,2304,
28, -1, -1, G_TEXT, NONE, NORMAL,
(LONG)&rs_tedinfo[4], 19,1801, 8,1,
29, -1, -1, G_TEXT, NONE, NORMAL,
(LONG)&rs_tedinfo[5], 275,3, 9,1,
30, -1, -1, G_BUTTON, 0x5, NORMAL,
(LONG)"Abbruch", 1558,270, 9,1,
31, -1, -1, G_TEXT, NONE, NORMAL,
(LONG)&rs_tedinfo[6], 1303,1540, 512,512,
36, 32, 35, G_BOX, NONE, NORMAL,
0xFF1100L, 1567,2563, 1291,3075,
33, -1, -1, G_TEXT, NONE, NORMAL,
(LONG)&rs_tedinfo[7], 260,1282, 6,1,
34, -1, -1, G_BOX, 0x51, OUTLINED,
0xFF1170L, 1281,2306, 257,2304,
35, -1, -1, G_TEXT, NONE, NORMAL,
(LONG)&rs_tedinfo[8], 260,3840, 5,1,
31, -1, -1, G_BOX, 0x51, OUTLINED,
0xFF1170L, 1281,513, 257,2304,
37, -1, -1, G_TEXT, NONE, NORMAL,
(LONG)&rs_tedinfo[9], 801,259, 8,1,
0, -1, -1, G_BUTTON, 0x27, NORMAL,
(LONG)"Ok", 1569,526, 265,3840,
-1, 1, 9, G_BOX, NONE, OUTLINED,
0x21100L, 0,0, 1827,3340,
2, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"Das erste Terminalprogramm",
1284,2052, 26,1,
3, -1, -1, G_BOXTEXT, NONE, OUTLINED,
(LONG)&rs_tedinfo[10], 1290,3584, 526,1,
4, -1, -1, G_TEXT, NONE, NORMAL,
(LONG)&rs_tedinfo[11], 1542,1538, 791,3584,
5, -1, -1, G_ICON, SELECTABLE, NORMAL,
(LONG)&rs_iconblk[0], 1026,3328, 4,1794,
6, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"für den Atari TT und Mega STE",
515,2309, 29,1,
7, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"mit bis zu 4 Schnittstellen ",
260,2566, 28,1,
8, -1, -1, G_IBOX, NONE, OUTLINED,
0x11100L, 2,3587, 288,2564,
9, -1, -1, G_TEXT, NONE, NORMAL,
(LONG)&rs_tedinfo[12], 1284,1289, 283,3072,
0, -1, -1, G_BUTTON, 0x27, NORMAL,
(LONG)"OK", 1292,3594, 1289,257,
-1, 1, 4, G_BOX, NONE, NORMAL,
0x1141L, 0,0, 45,20,
2, -1, -1, G_ICON, SELECTABLE, NORMAL,
(LONG)&rs_iconblk[1], 2,10, 263,1794,
3, -1, -1, G_ICON, SELECTABLE, NORMAL,
(LONG)&rs_iconblk[2], 2,13, 263,1794,
4, -1, -1, G_ICON, SELECTABLE, NORMAL,
(LONG)&rs_iconblk[3], 2,7, 519,1794,
0, -1, -1, G_ICON, 0x21, NORMAL,
(LONG)&rs_iconblk[4], 2,4, 519,1794,
-1, 1, 6, G_IBOX, NONE, NORMAL,
0x0L, 0,0, 90,25,
6, 2, 2, G_BOX, NONE, NORMAL,
0x1100L, 0,0, 90,513,
1, 3, 5, G_IBOX, NONE, NORMAL,
0x0L, 2,0, 25,769,
4, -1, -1, G_TITLE, NONE, NORMAL,
(LONG)" TT44TT ", 0,0, 8,769,
5, -1, -1, G_TITLE, NONE, NORMAL,
(LONG)" Datei ", 8,0, 7,769,
2, -1, -1, G_TITLE, NONE, NORMAL,
(LONG)" Optionen ", 15,0, 10,769,
0, 7, 21, G_IBOX, NONE, NORMAL,
0x0L, 0,769, 340,19,
16, 8, 15, G_BOX, NONE, NORMAL,
0xFF1100L, 2,0, 20,8,
9, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"  Zum Programm...   ", 0,0, 20,1,
10, -1, -1, G_STRING, NONE, DISABLED,
(LONG)"--------------------", 0,1, 20,1,
11, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"1", 0,2, 20,1,
12, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"2", 0,3, 20,1,
13, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"3", 0,4, 20,1,
14, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"4", 0,5, 20,1,
15, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"5", 0,6, 20,1,
7, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"6", 0,7, 20,1,
21, 17, 20, G_BOX, NONE, NORMAL,
0xFF1100L, 10,0, 18,4,
18, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"  Öffnen", 0,0, 18,1,
19, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"  Schließen", 0,1, 18,1,
20, -1, -1, G_STRING, NONE, DISABLED,
(LONG)"------------------", 0,2, 18,1,
16, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"  Ende", 0,3, 18,1,
6, 22, 27, G_BOX, NONE, NORMAL,
0xFF1100L, 17,0, 23,6,
23, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"  Parameter...", 0,0, 23,1,
24, -1, -1, G_STRING, NONE, DISABLED,
(LONG)"-----------------------",0,1,23,1,
25, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"  Zoomboxen", 0,2, 23,1,
26, -1, -1, G_STRING, NONE, DISABLED,
(LONG)"-----------------------",0,3,23,1,
27, -1, -1, G_STRING, NONE, NORMAL,
(LONG)"  INF-Datei laden...", 0,4, 23,1,
21, -1, -1, G_STRING, LASTOB, NORMAL,
(LONG)"  INF-Datei sichern...",0,5,23,1
};

#define NUM_OBS 81
#define NUM_TREE 4
```



STE-Soundbox

Teil 4 : Abspielen verschiedener Frames mittels Interrupt

Beim Erzeugen mehrerer Klänge oder Tonfolgen hintereinander tritt das Problem auf, daß die Übergänge zwischen den einzelnen Tönen sehr unsauber klingen, wenn man dabei immer denselben Frame verwendet.

Diese Problematik kann man umgehen, indem man die einzelnen Töne in verschiedene Frames packt und zwischen diesen umschaltet. Atari hat sich dazu ein recht elegantes Verfahren einfallen lassen.
Doch zunächst soll ein anderer Bereich kurz beleuchtet werden, nämlich die

Stereoklangerzeugung

Viele werden sich sicher schon gefragt haben, wie man es bewerkstelligt, auf jedem Stereokanal einen eigenen Klang oder eine eigene Tonfolge zu generieren. Daß man die Lautstärke der beiden Kanäle getrennt regeln kann, das kann doch nicht alles sein! Einige erinnern sich vielleicht noch an das 'Sound Mode Control'-Register, mit dem man in den Stereomodus schalten kann. Also frisch ans Werk und mit gesetzten 'Sound Mode Control'-Register einen Frame ausgegeben!

Leider ist es nicht ganz so einfach. Der Soundchip holt sich nämlich im Stereomodus die Daten 16-bitweise aus dem Frame. Das High-Byte des gelesenen Wortes wird dann dem linken Kanal zugordnet, das Low-Byte dem rechten Kanal. Man muß also dafür Sorge tragen, daß die Klangdaten der beiden Kanäle richtig in den Frame eingetragen werden.

Für diejenigen, die sich das zuerst an einem Beispiel ansehen wollen, ist das Listing 'STEREO.C' gedacht. Dort werden die beiden Kanäle mit unterschiedlichen Klängen angesteuert.

Rahmenumschaltung

So, und nun zurück zum eigentlichen Thema dieser Folge. Nehmen wir einmal folgende Aufgabenstellung als gegeben an: zwei Frames der Länge 1 und 2 Sekunden sollen nacheinander abgespielt werden. Dies läßt sich noch recht einfach bewerkstelligen.

Bevor der Soundchip mit dem Abarbeiten eines Frames beginnt, speichert er die Werte aus dem 'Frame Base Address'- und dem 'Frame End Address'-Register intern zwischen und beginnt dann erst mit der Tonausgabe. Das bedeutet, daß man, nachdem der erste Frame aktiviert ist, gleich die Adresse des zweiten Frames eintragen kann. Der Soundchip gibt den ersten Frame aus und registriert an dessen Ende, daß bereits ein neuer Frame eingetragen ist und startet diesen.

So weit, so gut - das obige Verfahren geht aber in die Hose, wenn man nach dem zweiten Frame noch einen dritten ausgeben möchte. Dann kann es passieren, daß der dritte Frame in die Register eingetragen wird, bevor der erste fertig ausgegeben ist. Damit würden jedoch die Daten des zweiten Frames überschrieben werden.

Langer Rede, kurzer Sinn - eine Möglichkeit muß her, mit der man feststellen kann, wann die Ausgabe eines Frames beendet ist. Dazu existiert am DMA-Soundchip ein Signal namens 'DMA Sound Active'. Dieses ist 1, wenn ein Frame ausgegeben wird, und 0, wenn gerade keine Ausgabe erfolgt. Das Signal ist mit dem externen Eingang von Timer A verbunden. Setzt man den Timer A in den Event-Count-Mode, und füllt man den Abwärtszähler von Timer A mit dem Wert 1, so passiert folgendes:

Während der Ausgabe eines Frames ist das Signal am Eingang von Timer A 1. Am Ende des Frames geht das Signal von 1 auf 0. Das veranlaßt den Abwärtszähler, seinen Wert um 1 auf 0 zu erniedrigen. Beim Wechsel von 1 auf 0 des Abwärtszählers generiert Timer A einen Interrupt. Genau dieser Interrupt wird dazu benutzt, zwischen den einzelnen Frames umzuschalten.

Um zu verhindern, daß durch die Zeit, die für das Eintragen der Frame-Adressen benötigt wird, bei einer Umschaltung zwischen den Frames eine Lücke entsteht, ist man bei Atari noch einen Schritt weiter

gegangen. Das 'DMA Sound Active'-Signal geht genau dann von 1 auf 0, wenn sich der Soundchip das letzte Byte aus dem Frame geholt hat. Da die Bytes aber intern in einem 64-Bit-FIFO-Speicher gepuffert werden, wird der Interrupt schon 8 Bytes vor Ende des Frames generiert. Die Interrupt-Routine hat folglich genauso lange Zeit, die Adressen des neuen Frames einzutragen, wie der Soundchip benötigt, 8 Bytes abzuspielen. Das sind im kritischsten Fall (bei 50066 Hz Sampling-Rate) 160 µs im Monomodus und 80 µs im Stereomodus.

Zur Auflockerung des ganzen folgt auch hier wieder ein Beispiel. Die Anwendung besteht dabei aus den Listings 'FRM.S', 'FRM.H' und 'FRAMES.C'. 'FRM.S' beinhaltet folgende von Turbo-C zugängliche Funktionen.

frm_stop schaltet die Tonausgabe ab und sperrt den Timer A-Interrupt. *frm_start* erwartet als Parameter einen Zeiger auf eine FRAME-Struktur. In dieser sind alle Frames zusammengefaßt, die nacheinander abgespielt werden sollen (siehe 'FRM.H'). *frm_start* installiert die Interrupt-Routine *f_next*, die der Umschaltung zwischen den Frames dient, und setzt den Timer A auf Ereigniszählung. Danach wird der erste Frame in die Soundchip-Register eingetragen. Der Rest, sprich die Umschaltung, läuft dann ganz von selbst ab.

In den Routinen ist noch eine kleine Besonderheit enthalten, nämlich die Möglichkeit, einen Frame nicht nur einmal, sondern mehrmals hintereinander abzuspielen. Dies geschieht, indem man den Abwärtszähler von Timer A mit der gewünschten Zahl an Wiederholungen füllt und das 'Sound DMA Control'-Register auf Wiederholung setzt. Timer A generiert jetzt den Interrupt erst dann, wenn sein Abwärtszähler auf 0 heruntergezählt hat.

Noch ein Wort zur FRAME-Struktur. Im Wert *anz_frames* muß die Anzahl der Frames stehen, zwischen denen umgeschaltet werden soll. Dieser Wert wird automatisch von der Funktion *frm_alloc* gefüllt. Des weiteren legt *frm_alloc* Speicherplatz für die Frames an. Jeder Frame besitzt eine eigene FSOUND-Struktur (siehe 'FRM.H'). Wie die Frames danach vom Programmierer zu füllen sind, kann man der Funktion *main* aus 'FRAMES.C' entnehmen. Auch hier ist bei eigenen Experimenten die Reihenfolge der Eintragungen und Funktionsaufrufe einzuhalten.

Übrigens, die Interrupt-Routine *f_next*, die für das Umschalten zuständig ist, benötigt für diese Aufgabe genau 64.5 µs, liegt also deutlich unter den für den worst case geforderten 80 µs.

Peter Engler

Literatur:

[1] STE Developer Addendum
[2] Jankowski, Reschke, Rabich: Atari ST Profibuch, Sybex 1989, Seite 699 ff.

```
/* ------------------------------------- */
/* --- STEREO.C : Stereo-Sound-Erzeugung
                  auf dem STE         --- */
/* --- (c) 1991 MAXON Computer        --- */
/* --- In Turbo-C 2.0 von Peter Engler - */
/* ------------------------------------- */



#include <stdio.h>
#include <stdlib.h>
#include <ext.h>
#include <tos.h>
#include <math.h>

#include "sout.h"


/* --- Prototypen der im Modul verwendeten
       Funktionen --- */

int snd_alloc( SOUND *, unsigned long );
void snd_free( SOUND * );
void stereo( SOUND * );
int main( void );



/* --- Anlegen des Arrays für die zu
       wandelnden Bytes --- */

int snd_alloc( SOUND *snd,
               unsigned long anz )
{
    /* --- Speicherplatz belegen --- */
    snd -> s_ptr = (char *)  malloc( anz );

    /* --- Fehler aufgetreten --- */
    if (! snd -> s_ptr)
    {
        snd -> anz_bytes = 0L;
        return( -1 );
    }

    /* --- Anzahl Bytes des reservierten
           Bereichs --- */
    snd -> anz_bytes = anz;

    /* --- Anzahl Bytes, die pro Sekunde
           gewandelt werden --- */
    switch( snd -> mode_reg & 0x000F )
    {

        case MOD_FR50K : snd ->
        bytes_pro_sekunde = 50066L;
                    break;
        case MOD_FR25K : snd ->
        bytes_pro_sekunde = 25033L;
                    break;
        case MOD_FR12K : snd ->
        bytes_pro_sekunde = 12517L;
                    break;
        case MOD_FR6K  : snd ->
        bytes_pro_sekunde = 6258L;
                    break;
        default        : snd ->
        bytes_pro_sekunde = 0L;
                    break;


    }


    return( 0 );
}



/* --- Freigeben des Arrays der SOUND-
       Struktur --- */

void snd_free( SOUND *snd )
{
    free( snd -> s_ptr );
}




/* --- Generieren der Werte (rechts Sinus,
       links Sägezahn ) --- */

void stereo( SOUND *snd)
{
```

→

```
    unsigned long bytes_pro_periode, index;
    char *h_ptr;

    h_ptr = snd -> s_ptr;

    /* --- Berechnung der Anzahl Bytes pro
           Periode --- */
    bytes_pro_periode = snd ->
    bytes_pro_sekunde / snd -> frequenz;

    /* -- Eintragen in SOUND-Struktur -- */
    snd -> anz_bytes =
                    bytes_pro_periode *2;


    /* --- Berechnung der Werte für den
           Vierklang --- */
    for (index = 0; index <
         bytes_pro_periode; index++)
    {
       /* --- Zuerst der linke Kanal --- */

       *h_ptr++ =
          (char) (127 * sin( 2.0 * M_PI *
          ((double) index) / (double)
          bytes_pro_periode) - 1);



       /* --- Dann der rechte Kanal --- */

       *h_ptr++ =
          (char) (255 * ((double) index)
          / ((double) bytes_pro_periode
          ) - 128);
    }
}




int main( )
{
    SOUND snd;

    /* --- 'mode_reg' immer vor 1. Aufruf
           von 'snd_alloc' setzen !! --- */
    snd.mode_reg = MOD_FR50K | MOD_STEREO;
    snd.control_reg = SND_IMMER;
    snd.frequenz = 220;  /* --- in Hz --- */

    /* --- Array f. den Frame anlegen --- */
    if (snd_alloc( &snd, 65536L))
        return(-1);



    /* --- Stereoklang berechnen --- */
     stereo( &snd );

    /* --- ... und abspielen --- */
    snd_play( &snd );

    /* --- Auf Tastendruck warten --- */
    (void)  getch( );

    /* --- Tonerzeugung ausschalten --- */
    snd_stop( );
    snd_free( &snd );

    return( 0 );
}
```

Manipulationen des RSC-Files

Teil 1

Haben Sie nicht auch schon einmal vergessen, das obligatorische RSC-File mit zu kopieren? Oder stört es Sie, daß das File, programmiert man unsauber, immer auf dem Laufwerk A gesucht wird?

Ist Ihnen der einfache String im Menü auch zu langweilig? Hätten Sie dort gern ein Bildchen? Oder wünschen Sie ein wenig Bewegung in der GEM-Dialogbox?

Wenn Sie alles mit „Ja" beantwortet haben, dann sollten Sie diesen und die folgenden Artikel lesen!

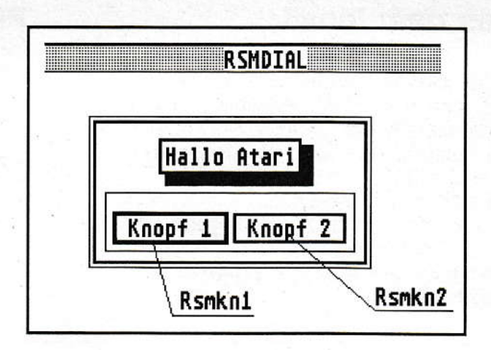
Über den Aufbau eines RSC-Files ist in den vergangenen Ausgaben von ST-Computer ausführlich berichtet worden. Lesen Sie bitte über den Kopf dieser Datei nach. Er enthält alle Zeiger auf die ausgewählten Objekttypen, welche wiederum, je nach Typ, Zeiger auf verwendete Strukturen wie *IconBlocks*, *BitBlocks*, *TedInfos* und Texte enthalten können. In diesen Blocks stehen dann wiederum Zeiger auf die dazugehörenden Texte, Grafiken und selbstdefinierte Programmteile.

Ein Unterschied ist bei den Zeigern zu beachten: Die Zeiger, gleichbedeutend mit Adressen, im Header sind 16 Bit breit. Damit lassen sich nur relative Entfernungen bis zu 32 KB nach vorn und nach hinten anspringen. Das ist auch ein Grund, weshalb eine RSC-Datei nicht größer als *MaxInteger*, 32 KB, sein kann (der andere Grund: die Länge steht im Header in 16 Bits. Damit wären allerdings 64 KB möglich).

Die Adressen in den Objekten und in den Blöcken sind jedoch 32 Bit breit. Folglich kann von dort aus der ganze Atari-Speicher erreicht werden. Dieser Umstand wird oft dazu genutzt, eigene Grafiken oder sog. *UserBlocks* erst nach dem Start des Programmes zu aktivieren. Von einem beliebigen Objekt, das mit dem RSC-Editor z.B. nur als I-Box erzeugt wurde, wird per Programm der Typ geändert und sein *Obj_Spec* als Adresse auf den eigenen Grafikblock umgebogen. Ich werde darauf beim Verändern eines Menüeintrages in ein Bildchen und beim Herumwandern des gleichen Bildes um eine Dialogbox zurückkommen. Nun zur Befriedigung der ersten, von Ihnen hoffentlich mit „Ja" beantworteten Frage:

Resourcen ohne Resource-Datei

Mit dem Resource-Editor wird ein einfacher Dialog (Bild) erzeugt. Die Ausgabe wird auf Pascal geschaltet. Es werden 2



Dateien abgelegt: eine Textdatei mit den von Ihnen ausgewählten Konstantennamen sowie die Resource-Datei mit der Extension *.RSC. Die Textdatei kann per Include-Anweisung in Ihr Programm eingebunden werden. Besser noch: Sie benennen diese Datei zur Unit um und übersetzen sie gesondert. Damit können Sie bei größeren Projekten von anderen Units auf sie zugreifen. Der Implementationsteil bleibt dabei leer.

Benennen Sie die einzelnen Objekte des Dialogbaumes mit den Namen, die Sie im Testprogramm (Listing 3) finden. Auf den Dialog werde ich noch einmal zur Demonstration von Bewegungen während des Dialogaufrufes zurückkommen. Die Konstanten habe ich hier der Bequemlichkeit halber direkt in das kleine Beispielprogramm (Listing 3) eingefügt. Die RSC-Datei wird nun mit dem Programm (Listing 1) in eine Unit mit Definitions- und Implementationsteil umgewandelt.

Der RSCMaker

Per Fileselektorbox wird eine RSC-Datei eingelesen. Dazu wird entsprechender Speicher angefordert. Diese Hex-Datei wird nun analysiert. Aus dem Kopf läßt sich ermitteln, wieviel Objekte vorhanden sind und bei welcher relativen Adresse (bezogen auf den Anfang der RSC-Datei) sie beginnen. Alle Werte werden im Definitionsteil für das gesonderte Hauptprogramm bereitgestellt.

Es reicht die Angabe der Adresse des ersten Objektes. Diesen Zeiger nenne ich in meiner GEM-Library *PtrObjTree*, einen Zeiger auf einen Objektbaum. Die weiteren Konstanten, Typen und Variablen können evtl. auch benötigt werden (ich brauchte sie allerdings noch nie).

Der Definition folgt die Implementation, das eigentliche Programm. Warum wird diese nun benötigt?

Es stellt zum einen dem Compiler alle Daten des RSC-Files als ASCII-Datei zur Übersetzung zur Verfügung. Beim Maxon-Pascal werden solche Daten mit ASM .. END geklammert, bei anderen Compilern sind es die Wörter INLINE oder

CODE. Diese Anweisungen bewirken, daß die folgenden ASCII-Werte wieder in Hexcode rückverwandelt werden, machen also das Gegenteil von dem, was ein Teil des RSC-Makers bewirkt.

Die Anweisung *IntToHex* ist in einer gesonderten Unit *NumConv* enthalten. Dort wird aus einer Integerzahl die Form $ABCD als ASCII-String erzeugt. Die Unit *NumConv* befindet sich auf der Diskette.

Der Implementationsteil reloziert zum anderen alle RSC-Daten, sofern es sich um 32-Bit-Adressen handelt.

Diese stellt der RSC-Editor nur als relative Adressen, bezogen auf den Dateianfang, bereit. Nach dem Start des Programmes, das die hier erzeugte Unit einbindet, wird zuerst die aktuelle absolute Adresse des Hex-Blockes ermittelt. Zu dieser werden dann die relativen Werte addiert. Das ist übrigens auch eine der Arbeiten der AES-Funktion *RscLoad()*.! Jene hat darüber hinaus aber noch die Aufgabe, Platz für die zu ladende RSC-Datei zu reservieren. Letzteres entfällt bei unserem Vorgehen, da sich die Datei ja bereits im Speicher befindet. Die 16-Bit-Adressen brauchen, da sie vom Dateianfang aus zeigen, nicht verändert zu werden.

Was aus Ihrer RSC-Datei geworden ist, sehen Sie im Listing 2.

Alle Kommentare und Absätze in der Prozedur rsdata habe ich nachträglich manuell eingetragen. Damit soll es Ihnen leichter gemacht werden, den Zusammenhang dieser Prozedur mit dem RSC-Bild zu erkennen. Suchen Sie Ihre Strings und wo deren Adressen stehen. Sehen Sie sich den Umweg über die Struktur *TEDINFO* beim Typ G_BOXTEXT an!

Eine allgemeine Anmerkung zum Listing: Wie Sie sicherlich erkannt haben, stimmen die GEM-Aufrufe nicht so ganz mit dem überein, was Sie aus dem Pascal-Handbuch kennen. Das Programm war ursprünglich in Megamax-Modula 2 geschrieben. Mir haben in dieser Sprache die einfacheren Aufrufe gefallen. So werden, wo möglich, keine Felder übergeben sondern Verbunde z.B für Punkte und Rechtecke. Auf der Diskette finden Sie die Units für AES und VDI als Quell-Listing. Wenn Sie diese nicht verwenden wollen: es sind nur wenige Aufrufe im Programm wie *InitGem*, *FormAlert* und *FileSelect*. Die Anpassung an die Original-Pascal-Units dürfte deswegen kein Problem darstellen.

In der nächsten Folge wird ein Bildchen (ein Hundchen, Stad-sei-Dank) in eine Menüleiste eingefügt.

Bruno Volkmer

Literatur:
Handbuch Maxon-Pascal
Handbuch Megamax Modula 2, ASH
Aumiller, Luda, Möllmann: GEM-Programmierung in C, Verlag Markt&Technik
Diverse Artikel in ST-Computer

```
(*************************************************
*                                                *
*                  Listing 1                     *
*                                                *
*  Programm zum Erzeugen einer Unit aus          *
*  RSC-Files, wie sie ein RSC-Editor erzeugt.    *
*  Erspart das jeweils zweite File m. der Exten-*
*  sion .RSC, das bei Verwendung von Dialogen,   *
*  Menus etc zwangsläufig zu jedem Programm      *
*  gehört und welches natürlich nie auf der      *
*  gleichen Ebene gefunden wird, von der das     *
*  Hauptprogramm gestartet wurde.                *
*  geschrieben in Megamax Modula 2, umgesetzt    *
*  in Maxon Pascal.                              *
*  Die Strings wurden wegen d. 49 Zeichen/Zeile *
*  stellenweise geteilt.                         *
*                                                *
*  (c) 1991 MAXON Computer                       *
*  Bruno Volkmer             1990/1991           *
*                                                *
*************************************************)

PROGRAM RSCMkPas;

(*$F+*)
Uses Dos, AES, VDI, GEM, NumConv, GrafBase;


TYPE tBuffer   = ^wordArray;
     wordArray = ARRAY [0..15999] OF INTEGER;
     ADDRESS   = POINTER;

VAR  handle              : DeviceHandle;
     inpath,inname,
     infile,egal,Pfad,
     outname, outpath,
     modname      : STRING;
     fhandle      : File;      (* Filehandler *)
     f            : TEXT;
     RscLaenge,
     gelesen      : INTEGER; (*  RSC-Länge   *)
     Buffer       : tBuffer; (* Puffer RSC   *)
     ok,success   : BOOLEAN; (* allg. VARs   *)
     nIcons, nObjects,     (* n = Anzahl der. *)
     nBitBlk, nFreeImg,    (* . Objekte eines *)
     nTrees, dummy,        (* .. Typs         *)
     nFreeStr, nTeds  : INTEGER;
     stat             : INTEGER; (* File-Stat*)
     iconsdrin        : BOOLEAN; (* Icons ? *)
     ImpStr,
     HexString        : String;  (* Hilfsst. *)
     but              : INTEGER;(* FormAlert *)

(*************************************************
* Einlesen des vom RSC-Editor erzeugten RSC-    *
* Files Vorgabeextension zum Lesen ist *.RSC.,  *
*************************************************)

FUNCTION RscFileRead:BOOLEAN;
BEGIN
     inpath := '\*.RSC';
     inname := '         .RSC';
     SelectFile(inpath,inname,ok);
     IF ok THEN BEGIN   (* wenn nicht abgebroch.*)
      modname := inname; (* Filename aufheben.  *)
      FSplit (inpath,inpath,egal,egal);
      inname := inpath + inname;
      (*$I-*)
      Reset(fhandle,inname);  (*  Datei öffnen  *)
      (*$I+*)
      IF IoResult = 0 THEN BEGIN
        RscLaenge := FileSize(fhandle);
        GetMem(Buffer,RscLaenge);
        BlockRead(fhandle,Buffer^,
                      RscLaenge,gelesen);
        Close(fhandle);      (* und Schließen   *)
        RscFileRead := TRUE
      END
     ELSE  RscFileRead := FALSE;
    END ELSE RscFileRead := FALSE;
END (* RscFileRead *);

(*************************************************
  Ausgabedatei zum Schreiben erzeugen und Öffnen.
  Es wird  der Name der RSC-Datei verwendet,
  wobei das RSC entfernt wird  und durch .PAS
  ersetzt wird.
*************************************************)


FUNCTION DateiOeffnen:BOOLEAN;
VAR posit:BYTE;
BEGIN
 outname := inname;
 posit := Pos ( 'RSC',outname); (* 'RSC' suchen *)
 IF posit > 0 THEN BEGIN     (* wenn gefunden.. *)
    Delete(outname,posit,3);(* 'RSC' löschen    *)
    outname := outname + 'PAS'; (* dafür 'PAS'
                                      anhängen *)
    Rewrite(f,outname);             (* neue Datei *)
 END;
   IF IoResult <> 0 THEN DateiOeffnen := FALSE
   ELSE DateiOeffnen := TRUE;      (* Fehler?    *)
END (* DateiOeffnen *);
```

→

```

(***********************************************
  Ab hier wird das Interfaceteil erzeugt,
  Objekttypen werden aus der Unit  Gem geholt.
***********************************************)

PROCEDURE MachDefKopf;
VAR posit : BYTE;
BEGIN
 WriteLn(f);
 posit := Pos('.',modname);
 Delete(modname,posit,4);
 Write(f,'Unit ');Write(f,modname);
 WriteLn(f,';');
 WriteLn(f); WriteLn(f);
 WriteLn(f,'(*Erzeugt vom Ressourcemacher 1.1*)');
 WriteLn(f,'(* B.Volkmer, 1991 *) ');
 WriteLn(f);
 Write(f,'(* Version des RessourceFiles : ');
 HexString := IntToHex(Buffer^[0]);
 WriteLn(f,HexString,' *)');

 WriteLn(f,'(*$F+*)');
 WriteLn(f,'INTERFACE');
 WriteLn(f); WriteLn(f,'Uses Gem, Bios;');
 WriteLn(f);

END (*MachDefKopf*);


PROCEDURE MachTypes;
BEGIN
 WriteLn(f);
 WriteLn(f,'TYPE');
 IF nObjects <> 0 THEN
  WriteLn(f,
  ' TreeArray     = ARRAY[0..nTrees] OF POINTER;');
 IF nFreeStr <> 0 THEN
  WriteLn(f,
  ' StringArray =ARRAY[0..nFreeStr] OF POINTER;');
 IF nFreeImg <> 0 THEN
  WriteLn(f,
  ' ImgArray    = ARRAY[0..nFreeImg] OF POINTER;');
 IF nTeds <> 0 THEN
  WriteLn(f,
  ' TedInfoArray = ARRAY[0..nTeds]  OF TEdInfo;');
 IF nIcons <> 0 THEN
  WriteLn(f,
  'IconBlkArray =ARRAY[0..nIcons] OF IconBlock;');
 IF nBitBlk <> 0 THEN
  WriteLn(f,
  ' BitBlkArray = ARRAY[0..nBits]  OF BitBlock;');
 IF nObjects <> 0 THEN
  WriteLn(f,
  'ObjAddrArray = ARRAY[0..nObjects] OF Object;');
 WriteLn(f);

END  (*MachTypes *);



(***********************************************
 Erzeugung der Konstanten im Interfaceteil
***********************************************)

PROCEDURE MachDefConst;

   PROCEDURE MachConst(txt:String;
                        point,flag:INTEGER;
                         VAR nX:INTEGER);

   VAR wert  : INTEGER;
       stri  : String;
   BEGIN
        wert := Buffer^[point];
       nX   := wert;
       flag := Buffer^[flag];
       IF flag <> 0 THEN BEGIN
         Str(wert:5,stri);
         WriteLn(f,' ',txt,stri,';');
       END;
   END (*MachConst*);

BEGIN    (* MachDefConst *)

 WriteLn(f);
 WriteLn(f,'CONST');
 MachConst('nObjects = ',10,10,nObjects);
 MachConst('nTrees   = ',11,11,nTrees);
 MachConst('nTeds    = ',12,12,nTeds);
 MachConst('nIcons   = ',13,13,nIcons);
 MachConst('nBits    = ',14,14,nBitBlk);
 MachConst('nFreeStr = ',15,15,nFreeStr);
 MachConst('nFreeImg = ',16,16,nFreeImg);

 MachConst('xObjects = ',1,10, dummy);
 MachConst('xTrees   = ',9,9,  dummy);
 MachConst('xTEds    = ',2,12, dummy);
 MachConst('xIcons   = ',3,13, dummy);
 MachConst('xBits    = ',4,14, dummy);
 MachConst('xFreeStr = ',5,15, dummy);
 MachConst('xFreeImg = ',8,16, dummy);

END (* MachDefConst *);


(***********************************************
  Erzeugung der Variablen im Interfaceteil
***********************************************)

PROCEDURE MachDefVars;
BEGIN
 WriteLn(f);
 WriteLn(f,'VAR');
 IF nObjects <> 0 THEN
   WriteLn(f, ' TreeAddr    : ^TreeArray;');

 IF nFreeStr <> 0 THEN
  WriteLn(f, ' StringAddr  : ^StringArray;');

 IF nFreeImg <> 0 THEN
  WriteLn(f, ' ImgAddr       : ^ImgAddrArray;');

 IF nTeds <> 0 THEN
  WriteLn(f, ' TedInfoAddr : ^TedInfoArray;');

 IF nIcons <> 0 THEN
  WriteLn(f, ' IconBlkAddr : ^IconBlkArray;');

 IF nBitBlk <> 0 THEN
  WriteLn(f, ' BitBlkAddr  : ^BitBlkArray;');

 IF nObjects <> 0 THEN
  WriteLn(f, ' ObjectAddr    : ^ObjAddrArray;');
 WriteLn(f);

END  (*MachDefVars *);


(***********************************************
  ab hier wird die Implementation erzeugt.
***********************************************)

PROCEDURE MachImpKopf;
BEGIN

WriteLn(f); WriteLn(f,'IMPLEMENTATION');
WriteLn(f);
END  (*MachImpKopf *);

(***********************************************
  und ein paar Konstante in der Implementation
***********************************************)

PROCEDURE MachImpConst;
BEGIN
WriteLn(f);
WriteLn(f,'CONST');
WriteLn(f,'  GBOX  = 20; USERBLK  = 24;');
WriteLn(f,'  GIBOX = 25; GBOXCHAR = 27;');
WriteLn(f);

IF (Buffer^[13] <> 0)  OR (Buffer^[14] <> 0) THEN
BEGIN
   iconsdrin := TRUE;
   WriteLn(f,'  GIMAGE = 23; GICON = 31;');
 END;
END  (*MachImpConst *);

(***********************************************
  ein paar Variable gibts auch im Imp.Teil
***********************************************)

PROCEDURE MachImpVars;
```

→

```
BEGIN
 WriteLn(f); WriteLn(f,'VAR');
 WriteLn(f,'  i                  : INTEGER;');
 WriteLn(f,'  chW,chH,rezol      : INTEGER;');
 WriteLn(f,'  RSDATA             : POINTER;');
 WriteLn(f);
END  (*MachImpVars *);


(**************************************************
  Prozedur liest Hexcodes der RSC-datei aus dem
  Buffer,  in den die Datei geladen wurde.
  Sie erzeugt dann ASCII- Code und formatiert ihn.
**************************************************)


PROCEDURE MachASCIIausHex;
                    (* der RSC-Daten als Hexcode *)
VAR i,max,w : INTEGER;
    ok      : BOOLEAN;
BEGIN
    max := gelesen - 1;
    WriteLn(f);
    WriteLn(f,'VAR rsdta : POINTER;');
    WriteLn(f,'PROCEDURE rscdata;');
    WriteLn(f,'  ASSEMBLER;');
    WriteLn(f,'  ASM');
    WriteLn(f,'    LEA  @start,A0');
    WriteLn(f,'    MOVE.L A0,rsdta');
    WriteLn(f,'    BRA  @fertig'); ;
    WriteLn(f,'@start:');

    max := max DIV 2;
    FOR i := 0 TO max DO BEGIN
      w := Buffer^[i];
      HexString := IntToHex(w);
      IF i MOD 8 = 0 THEN
         HexString := '    DC.W '+HexString;
      Write(f,HexString);
      IF (i+1) MOD 8 = 0 THEN WriteLn(f)
        ELSE BEGIN
        IF i + 1 > max THEN ELSE
        Write(f,',');
      END;
    END;
    WriteLn(f);
    WriteLn(f,'@fertig:');
    WriteLn(f,' END;');   WriteLn(f);
END  (*MachASCIIausHex *);


(**************************************************
 Hier wird das eigentliche Programm zusammen-
 gestellt
**************************************************)

PROCEDURE  MachPgm;
BEGIN
WriteLn(f); WriteLn(f);
WriteLn(f,'BEGIN');
WriteLn(f,' rscdata;');
IF nTrees <> 0 THEN BEGIN
 WriteLn(f,
  '(*Liste der Objektbaum-Adressen relozieren*)');
 WriteLn(f);
 WriteLn(f,' RSDATA := rsdta;');
 WriteLn(f,' TreeAddr := POINTER(LONGINT(RSDATA)',
                       '+LONGINT(xTrees));');
 WriteLn(f,' FOR i := 0 TO nTrees-1 DO');
 WriteLn(f,'   TreeAddr^[i] := POINTER(LONGINT',
             '(TreeAddr^[i])+LONGINT( RSDATA));');
END;

IF nTeds <> 0 THEN BEGIN
 WriteLn(f,
 '(* Adressen innerhalb  TEDinfos reolzieren *)');
 WriteLn(f,
          'TedInfoAddr := POINTER(LONGINT',
                    '(RSDATA)+LONGINT(xTEds));');
 WriteLn(f,'  FOR i := 0 TO nTeds - 1  DO ');
 WriteLn(f,'   WITH TedInfoAddr^[i] DO BEGIN');
 WriteLn(f,'      validPtr := POINTER(LONGINT',
                 '(validPtr)+LONGINT(RSDATA));');
 WriteLn(f,'      tmpltPtr := POINTER(LONGINT',
                 '(tmpltPtr)+LONGINT(RSDATA));');
 WriteLn(f,'      textPtr  := POINTER(LONGINT',
                  '(textPtr)+LONGINT(RSDATA));');
 WriteLn(f,'   END;');
END;

IF nIcons <> 0 THEN BEGIN
 WriteLn(f,
       '(* Adressen in IconBlocks relozieren *)');
 WriteLn(f,'  IconBlkAddr := POINTER(LONGINT',
                   '(RSDATA)+LONGINT(xIcons));');
 WriteLn(f,'  FOR i := 0 TO nIcons - 1  DO ');
 WriteLn(f,'   WITH IconBlkAddr^[i] DO BEGIN');
 WriteLn(f,'     data := POINTER(LONGINT(data)+',
                          'LONGINT(RSDATA));');
 WriteLn(f,'     mask := POINTER(LONGINT(mask)+',
                          'LONGINT(RSDATA));');
 WriteLn(f,'     txt := POINTER(LONGINT(txt)+',
                          'LONGINT(RSDATA));');
 WriteLn(f,'   END;');
END;

IF nBitBlk <> 0 THEN BEGIN
 WriteLn(f);
 WriteLn(f,
       '(* Adressen in BitBlocks relozieren *)');
 WriteLn(f,'  BitBlkAddr := POINTER(LONGINT',
                   '(RSDATA)+LONGINT(xBits));');
 WriteLn(f,'  FOR i := 0 TO nBits - 1  DO ');
 WriteLn(f,'   WITH BitBlkAddr^[i] DO');
 WriteLn(f,'     data := POINTER(LONGINT(data)',
                          '+LONGINT(RSDATA));');
END;

IF nObjects <> 0 THEN BEGIN
 WriteLn(f);
 WriteLn(f,
'(* Object-Specs relozieren, falls notwendig *)');
 WriteLn(f,'  ObjectAddr := POINTER(LONGINT',
                 '(RSDATA)+LONGINT(xObjects));');
 WriteLn(f,'  FOR i := 0 TO nObjects-1 DO BEGIN');
 WriteLn(f,'  WITH ObjectAddr^[i] DO BEGIN');
 WriteLn(f,'    IF (typ <> GBOX)  AND ',
                     '(typ <> GIBOX) AND');
 WriteLn(f,'    (typ <> USERBLK) AND');
 WriteLn(f,'    (typ <> GBOXCHAR) THEN');
 WriteLn(f,'    spec.more := POINTER(LONGINT',
               '(spec.more)+LONGINT(RSDATA));');
 WriteLn(f,'    chW := 8; chH := 16;');
 WriteLn(f,'    rezol := GetRez;');
 WriteLn(f,'    IF rezol < 2 THEN chH := 8;');
 WriteLn(f,'    WITH space DO BEGIN');
 WriteLn(f,'      y:=(y MOD 256) * chH + ',
                          '(y DIV 256);');
 WriteLn(f,'      x:=(x MOD 256) * chW + ',
                          '(x DIV 256);');
 WriteLn(f,'      h:=(h MOD 256) * chH + ',
                          '(h DIV 256);');
 WriteLn(f,'      w:=(w MOD 256) * chW + ',
                          '(w DIV 256);');

 WriteLn(f,'    END;');
 WriteLn(f,'  END;');
 WriteLn(f,' END;');
END;

IF nFreeStr <> 0 THEN BEGIN
 WriteLn(f,
 '(* Addressen der freien Strings relozieren *)');
 WriteLn(f,' StringAddr := POINTER(LONGINT',
                '(RSDATA)+LONGINT(xFreeStr));');
 WriteLn(f,'  FOR i := 0 TO nFreeStr - 1 DO');
 WriteLn(f,'    StringAddr^[i] := POINTER(LONGINT',
          '(StringAddr^[i])+LONGINT(RSDATA));');
END;

IF nFreeImg <> 0 THEN BEGIN
 WriteLn(f,
      '(* Adressen der FreeImages relozieren *)');
 WriteLn(f,' ImgAddr := POINTER(LONGINT(RSDATA)+',
                        'LONGINT(xFreeImg));');
 WriteLn(f,'  FOR i := 0 TO nFreeImg - 1 DO');
 WriteLn(f,'    ImgAddr^[i] := POINTER(LONGINT',
             '(ImgAddr^[i])+LONGINT(RSDATA));');
END;

END (* MachPgm *);

(**************************************************
   weils oben   auch so aussieht
```

→

```
****************************************************)

PROCEDURE MachImpEnde;
BEGIN
WriteLn(f,'END.');
END (* MachImpEnde *);

(****************************************************
    Datei schließen
****************************************************)

PROCEDURE DateiSchliessen;
BEGIN
 Close(f);
END (* DateiSchliessen *);


BEGIN (************ Hauptprogramm *************
***********)

  InitGem(RC,handle,ok);
         (* notwendig wegen F.Selektor-Box *)
  IF ok THEN BEGIN
     IF GemError = 0 THEN BEGIN
      iconsdrin := FALSE;
      but := 1;
      REPEAT
       IF RscFileRead THEN BEGIN
                  (* RSC-Datei aussuchen *)
         IF  DateiOeffnen THEN BEGIN;
                  (* und ggfs einlesen   *)
           MachDefKopf;
               (* Kopf Interfaceteil erzeugen *)

           MachDefConst; (* Konstanten eintragen*)
           MachTypes;
           MachDefVars;  (* Variablen eintragen *)
           MachImpKopf;  (* Kopf Impl. erzeugen *)
           MachImpConst; (* Konstanten eintragen*)
           MachImpVars;  (* Variablen hinzufügen*)
           MachASCIIausHex;
                (* RSC-Datei als INLINE-Proz.  *)
           MachPgm; (* das eigentliche Programm *)
           MachImpEnde;  (* Impl.Teil schließen *)
           DateiSchliessen;
         END
       ELSE
        FormAlert(1,
  '[3][Ausgabedatei läßt sich|nicht öffnen][Pech]'
            ,but)
       END
     ELSE
      FormAlert(1,
  '[3][Fehler bei der|Eingabe][Abbruch|nochmal]'
           ,but);

     UNTIL but = 1; (* abbruch *)
    END;
    ExitGem(handle);
  END;
END (* RSCMkPas *).

(****************************************************
```

```
(****************************************************
*                                                  *
*                   Listing 2                      *
*                                                  *
*  Die vom RSC-Maker erzeugte Unit.                *
*                                                  *
* Keine Angst, diesen Text brauchen Sie nicht      *
* einzutippen!                                     *
* Das erledigt der RSC-Maker für Sie!              *
*                                                  *
* Das Listing wurde mit Hand nachbearbeitet, um    *
* die Zusammenhänge zu verdeutlichen und um die    *
* Vorgabe, 49 Zeichen pro Zeile, einzuhalten       *
*                                                  *
****************************************************)

Unit RSMTEST;

(* Erzeugt vom Ressourcemacher 1.01 *)
(* B.Volkmer, 1991 *)

(* Version des RessourceFiles : $0001 *)
(*$F+*)
INTERFACE

Uses Gem, Bios;
                    (* Gem-Unit auf Diskette *)
                    (* darin nur Typdeklarationen *)
CONST
  nObjects =    5;
  nTrees   =    1;
  nTeds    =    1;
  xObjects =   94;
  xTrees   =  214;
  xTEds    =   66;

TYPE
 TreeArray     = ARRAY[0..nTrees]   OF POINTER;
 TedInfoArray = ARRAY[0..nTeds]    OF TEdInfo;
 ObjAddrArray = ARRAY[0..nObjects] OF Object;


VAR
 TreeAddr      : ^TreeArray;
 TedInfoAddr : ^TedInfoArray;
 ObjectAddr  : ^ObjAddrArray;

IMPLEMENTATION

CONST
  GBOX  = 20; USERBLK  = 24;
  GIBOX = 25; GBOXCHAR = 27;

VAR
  i                  : INTEGER;
  chW,chH,rezol      : INTEGER;
  RSDATA             : POINTER;


VAR rsdta : POINTER;
PROCEDURE rscdata;
  ASSEMBLER;
  ASM
   LEA  @start,A0
   MOVE.L A0,rsdta
   BRA  @fertig
@start:
     DC.W $0001          (* Version         *)
     DC.W $005E          (* Zeiger auf 1. Objekt  *)
     DC.W $0042          (* zeigt auf 1. TEDINFO  *)
     DC.W $005E          (* ^ auf 1. IconBlock,   *)
     DC.W $0042,$0042    (* BitBl. und fr. Strings*)
     DC.W $0024          (* zeigt auf 1. String   *)
     DC.W $0042,$0042    (* Image, freie Image, nix*)
     DC.W $00D6          (* Zeiger auf Baumadressen*)
     DC.W $0005          (* Anzahl der Objekte    *)
     DC.W $0001          (* Anzahl der Bäume      *)
     DC.W $0001          (* Anzahl der TEDINFOs   *)
     DC.W $0000,$0000,$0000,$0000      (* Rest,nix*)
     DC.W $00DA          (* Länge der RSC-Datei   *)

                         (* Text "Hallo Atari" *)
  (* 24 *) DC.W $4861,$6C6C,$6F20,$4174H
           DC.W $6172,$6900

  (* 30 *) DC.B $00                (* Maske     *)
  (* 31 *) DC.B $00                (* Valid     *)
  (* 32 *) DC.W $4B6E,$6F70,$6620,$3100
                                   (* "Knopf 1"*)
  (* 3A *) DC.W $4B6E,$6F70,$6620,$3200
                                   (* "Knopf 2"*)

  (* TEDINFO *)

  (* 42 *) DC.W $0000,$0024  (* Zeiger Text  *)
           DC.W $0000,$0030  (* Zeiger Maske *)
           DC.W $0000,$0031  (* Zeiger Valid *)
           DC.W $0003        (* Font IBM     *)
           DC.W $0006        (* reserviert   *)
           DC.W $0000        (* Just. links  *)
           DC.W $1180        (* Farbe        *)
           DC.W $0000        (* reserviert   *)
           DC.W $FFFC (* Rahmendicke -4,außen*)
           DC.W $000C        (* Länge Text,12*)
           DC.W $0001        (* Länge Maske,1*)

  (* 1. Objekt beginnt bei 5E *)
           (* next, head, tail *)
```

→

```
    (* 5E *) DC.W $FFFF,$0001,$0002
             DC.W $0014       (* Typ 20 = G_BOX*)
             DC.W $0000,$0010    (* flag, state*)
             DC.W $0002,$1100          (* spec*)
             DC.W $0001,$0001,$0017,$0006
                                      (*Maße*)

    (* 76 *) DC.W $0002,$FFFF,$FFFFH
             DC.W $0016    (* Typ 22 = G_BOXTEXT *)
             DC.W $0000,$0030   (* flags, state *)
             DC.W $0000,$0042   (* Adr. TedInfo *)
             DC.W $0006,$0001,$000B,$0001H

    (* 8E *) DC.W $0000,$0003,$0004
             DC.W $0019      (* Typ 25 = G_IBOX *)
             DC.W $0000,$0000    (* flags,state *)
             DC.W $0001,$1100          (* spec *)
             DC.W $0001,$0003,$0015,$0802

    (* A6 *) DC.W $0004,$FFFF,$FFFF
             DC.W $001A     (* Typ 26 = G_BUTTON *)
             DC.W $0017,$0000    (* flags, state *)
             DC.W $0000,$0032   (* Stringadr.    *)
             DC.W $0001,$0001,$0009,$0001

    (* BE *) DC.W $0002,$FFFF,$FFFF
             DC.W $001A     (* Typ 26 = G_BUTTON *)
             DC.W $0035,$0000    (* flags, state *)
             DC.W $0000,$003A     (* Stringadr. *)
             DC.W $000B,$0001,$0009,$0001

    (* D6 *) DC.W $0000,$005E    (*^auf 1.Objekt *)
@fertig:
END;


BEGIN            (***** Hauptprogramm ********)
  rscdata;            (* rsdata bekannt machen *)
(* Liste der Objektbaum-Adressen relozieren *)
  RSDATA := rsdta;
  TreeAddr
      := POINTER(LONGINT(RSDATA)+LONGINT(xTrees));
  FOR i := 0 TO nTrees-1 DO
    TreeAddr^[i]:= POINTER(LONGINT(TreeAddr^[i])
                         +LONGINT( RSDATA));
(* Adressen innerhalb der TEDinfos reolzieren *)
TedInfoAddr := POINTER(LONGINT(RSDATA)
                         +LONGINT(xTEds));
  FOR i := 0 TO nTeds - 1 DO
   WITH TedInfoAddr^[i] DO BEGIN
     validPtr := POINTER(LONGINT(validPtr)
                         +LONGINT(RSDATA));
     tmpltPtr := POINTER(LONGINT(tmpltPtr)
                         +LONGINT(RSDATA));
     textPtr  := POINTER(LONGINT(textPtr)
                         +LONGINT(RSDATA));
   END;

(* Object-Specs relozieren, falls notwendig *)
ObjectAddr := POINTER(LONGINT(RSDATA)
                         +LONGINT(xObjects));
  FOR i := 0 TO nObjects-1 DO BEGIN
    WITH ObjectAddr^[i] DO BEGIN
      IF (typ <> GBOX)  AND (typ <> GIBOX)  AND
         (typ <> USERBLK) AND
         (typ <> GBOXCHAR) THEN
          spec.more := POINTER(LONGINT(spec.more)
                                +LONGINT(RSDATA));
      chW := 8; chH := 16;
      rezol := GetRez;
      IF rezol < 2 THEN chH := 8;
      WITH space DO BEGIN
        y:=(y MOD 256) * chH + (y DIV 256);
        x:=(x MOD 256) * chW + (x DIV 256);
        h:=(h MOD 256) * chH + (h DIV 256);
        w:=(w MOD 256) * chW + (w DIV 256);
      END;
    END;
  END;
END.

(***********************************************)
```

```
(***********************************************
*                  Listing 3                   *
*  kleines Beispielprogramm zum Testen des     *
*  RSCMaker - Ergebnisses                      *
***********************************************)


PROGRAM RsmBeispiel;

USES VDI, AES, GEM, GrafBase, RSMTEST;

(* Die Konstantendatei des RSC-Editors wird hier
   der Übersichtlichkeit wegen gleich eingefügt *)

CONST
  Rsmdial   =    0;  (* Formular/Dialog *)
  Rsmtext   =    1;  (* BOXTEXT in Baum RSMDIAL *)
  Rsmkn1    =    3;  (* BUTTON in Baum RSMDIAL *)
  Rsmkn2    =    4;  (* BUTTON in Baum RSMDIAL *)

VAR
     handle  : INTEGER;
     ok      : BOOLEAN;
     baum    : PtrObjTree;
     ecke    : Rectangle;
     Button  : INTEGER;

BEGIN
   InitGem(RC,handle,ok);
   IF ok THEN BEGIN
      IF GemError = 0 THEN BEGIN
        baum := TreeAddr^[Rsmdial];
        FormCenter(baum, ecke);
        FormDial(reserveForm,ecke,ecke);
        DrawObject(baum,0,8,ecke);
        FormDo(baum,0,Button);
        FormDial(freeForm,ecke,ecke);
      END;
      ExitGem(handle);
   END;
END (* RsmBeispiel *) .

(***********************************************)
```

Memwatch



Modifikation kommt außer Mode

Schreiben Sie selber Programme für Atari ST und TT? Wenn ja, beachten Sie doch hoffentlich die Grundregeln der sauberen Programmierung. Dazu zählt halt das Übliche: nur dokumentierte Systemvariablen verwenden, keine direkten Bildschirmzugriffe usw. Man kennt das ja.

Als Programmierer sollte man jedoch auch darauf achten, insofern sauber zu arbeiten, als daß kein selbstmodifizierender Code verwendet wird. Bei Prozessoren, die wie der 68030 des TT einen internen Cache besitzen, sind andernfalls Fehler während der Programmausführung nicht auszuschließen. Probleme gibt es genau dann, wenn der Inhalt des Caches nicht mit dem des Hauptspeichers übereinstimmt und der Prozessor für die Bearbeitung des nächsten Befehls die Daten im Cache verwendet. Diese sind unter Umständen nicht mehr auf dem neuesten Stand, falls die korrespendierenden Bereiche im Hauptspeicher zwischenzeitlich modifiziert wurden. Ein Programmabsturz ist somit mehr als wahrscheinlich.

Die obige Situation kann eintreten, wenn sich ein Programm selbst modifiziert, ohne anschließend den Cache zu löschen. Liegt ein Teil des modifizierten Programmcodes direkt hinter der Stelle, auf die der Programmzähler zeigt, so befinden sich die inzwischen ungültigen Daten möglicherweise bereits im Cache und sorgen im weiteren Programmverlauf für einen Fehler. Wenn es sich wirklich nicht umgehen läßt, das laufende Programm zu modifizieren, dann muß nach einer solchen Manipulation unbedingt der Cache gelöscht werden.

Programme, die auf einem 68040-Prozessor laufen sollen, werden es übrigens besonders schwer haben. Der MC68040 kann seine beiden Caches (Daten- und Befehls-Cache) im sogenannten „Copy Back"-Modus einsetzen. In dieser Betriebsart werden neue Daten nicht sofort in den Speicher geschrieben, sondern nur bei Bedarf. Verändert sich ein Programm auf dem 68040 selbst, so hat dies zunächst lediglich Auswirkungen auf den Daten-Cache. Der Prozessor holt sich seine Befehle jedoch aus dem Befehls-Cache, der von selbstmodifizierenden Programmen nicht beeinflußt wird und somit noch den nicht modifizierten Code enthält ...

An der Wurzel gepackt

Nun sollte man es sich aber nicht so einfach machen, bei Verwendung von selbstmodifizierendem Code lediglich den Cache zu invalidieren und es dabei bewenden zu lassen. Dies stellt schließlich nur eine Notlösung dar. Gerade durch das Vorhandensein des Caches erreicht der 68030 seine hohe Rechenleistung. Wird der Cache gelöscht, sinkt der Datendurchsatz, was natürlich nicht wünschenswert ist.

Programme, die sich selbst modifizieren, bringen noch weitere Nachteile mit sich. Was geschieht, wenn ein solches Programm in ein Eprom gebrannt wird? Klar, daß beim Brennen selber natürlich nichts weiter passiert. Allerdings darf man sich anschließend nicht wundern, daß man mit seinem Eprom wenig anfangen kann, denn hier läßt sich nun gar nichts mehr modifizieren. Nun dürfte man beim ST oder TT nicht allzu häufig das Verlangen verspüren, Programme in Eproms zu verewigen. Schließlich lassen sich über den ROM-Port lediglich 128 kByte ROM ansprechen, was nicht gerade viel ist, wenn man sich den Umfang heutiger Programme vor Augen führt. Aber gerade für kleinere residente Routinen wie RAM-Disks oder Gerätetreiber kann der Einsatz von Eproms durchaus sinnvoll sein.

Schließlich noch ein Blick in die Zukunft. Wer garantiert, daß es nicht einmal eine TOS-Version geben wird, die das TEXT-Segment einer Applikation gegen Schreibzugriffe schützt? Gerade im Hinblick auf die Tendenz, mehrere Programme parallel laufen zu lassen, erscheint ein solcher Schutzmechanismus durchaus sinnvoll. Ein fehlerhaftes Programm wäre dann nicht mehr so schnell in der Lage, ein anderes Programm zum Absturz zu bringen. Die Betriebssicherheit eines Multitasking-Systems ließe sich somit deutlich erhöhen.

Es gibt also gute Gründe, die dafür sprechen, grundsätzlich auf selbstmodifizierenden Code zu verzichten. In den weitaus meisten Fällen ist dies mit durchaus vertretbarem Aufwand möglich. Oft genügt es, einen kleinen Programmteil aus dem TEXT-Segment in das DATA- oder BSS-Segment zu kopieren und dort zu verändern. Hierzu später mehr.

Das TEXT-Segment schützen

Es hätte zwar etwas für sich, wenn das TT-TOS Schreibzugriffe auf den Programmcode einer Applikation verhindern würde,

aber dies ist (noch) nicht der Fall. Zum jetzigen Zeitpunkt wäre ein solches Verhalten des TOS auch absolut nicht wünschenswert, denn es gibt eine ganze Reihe von Programmen, die ihr TEXT-Segment verändern. Programme, die für den Atari ST gedacht waren, brauchten sich schließlich nicht um solche Feinheiten zu kümmern.

Auch wenn TOS selbstmodifizierenden Code nicht verhindert, muß es nicht dabei bleiben, lediglich über die Frage zu spekulieren, inwiefern ein Programm sich selbst modifiziert. Die Hardware des TT bietet ideale Voraussetzungen, sich Gewißheit hierüber zu verschaffen. Mit seiner PMMU stellt der 68030-Prozessor alle Funktionen zur Verfügung, die benötigt werden, um einen Schreibschutz von TEXT-Segmenten zu realisieren.

Daß es die MMU prinzipiell erlaubt, einen Speicherbereich gegen Zugriffe zu schützen, wurde bereits in [1] verdeutlicht. Hier ging es jedoch um einen Schutz für den Bildschirmspeicher, der sich nicht nur auf Schreib-, sondern auch auf Lesezugriffe erstreckte. Für unseren Zweck brauchbar ist ein solches Verfahren nicht, da Lesezugriffe auf den Programmcode weiterhin zulässig sein sollen. Es stellt sich also die Frage, wie man es anstellen kann, ausschließlich Schreibzugriffe auf das TEXT-Segment zu unterbinden.

Wer schreibt, der bleibt?

Wirft man einen Blick auf den Aufbau der Seitendeskriptoren der 68030-MMU, so läßt sich bereits ablesen, welcher Weg gegangen werden kann. Das W-Bit erlaubt es, einzelne Speicherseiten gegen Änderungen zu schützen. Es kommt also darauf an, für jedes Programm, das auf dem TT gestartet wird, genau in den Deskriptoren das W-Bit zu setzen, die für die Adreßübersetzung des TEXT-Segments zuständig sind.

Eigentlich wäre es angebracht, Änderungen im Programmcode sowohl für User- als auch für den Supervisor-Zugriff zu unterbinden. Dies hätte auf dem TT jedoch fatale Folgen, die eng mit der ST-Kompatibilität zusammenhängen. Durch einen Design-Fehler beim 68000 ist es bei diesem Prozessor erlaubt, Befehle des Typs *MOVE SR,Dn* im User-Modus zu verwenden. Eigentlich sollte es in dieser Betriebsart nicht gestattet sein, auf Daten zuzugreifen, die dem Supervisor vorbehalten sind. Obiger Befehl erlaubt es nämlich, das System-Byte des Statusregisters auch im User-Modus auszulesen. Alle Prozessoren seit dem 68010 erlauben einen Zugriff auf den kompletten Prozessorstatus deshalb nur noch im Supervisor-Modus. Um die Prozessor-Flags (Condition Code Byte) anzusprechen, existieren neue Befehle wie *MOVE CCR,Dn*. Da diese beim 68000 jedoch nicht implementiert sind, ist man gezwungen, bei diesem Prozessor weiterhin das komplette Statusregister anzusprechen. Diese Operation führt auf den restlichen Motorola-Prozessoren jedoch zu einer Exception-Behandlung aufgrund einer Privilegverletzung.

CCR statt SR

Das TOS des TT löst dieses Problem, indem innerhalb der Exception-Routine Befehle wie *MOVE SR, Dn* in *MOVE CCR, Dn* modifiziert werden. Das TEXT-Segment eines Programms wird also vom Betriebssystem verändert! Auf diese Weise ist es möglich, Programme, die für einen 68000-Prozessor geschrieben wurden, ohne Änderungen auch auf einem 68030 ablaufen zu lassen. Die bei TOS gewählte Methode birgt den Vorteil, daß jeder Zugriff auf das System-Byte des Statusregisters nur ein einziges Mal zu einer Exception führt. Beim erneuten Durchlaufen des Codes ist das SR bereits durch CCR ersetzt, und es tritt nicht nochmals ein Fehler auf. (Wie es sich gehört, wird nach einer solchen Manipulation übrigens der Cache gelöscht.)

Wenn also TOS unter gewissen Bedingungen Eingriffe in das TEXT-Segment eines Programms vornimmt, müssen wir zulassen, daß der Programmcode aus dem Supervisor-Modus heraus verändert werden darf. Damit ist natürlich nicht mehr möglich, selbstmodifizierenden Code dann zu entlarven, wenn dieser im Supervisor-Modus abläuft. Die weitaus meisten Programme beschränken sich jedoch korrekterweise auf den User-Modus, so daß wir in der Regel durchaus feststellen können, ob ein Programm sich selbst modifiziert. Nur Programme, die ständig im Supervisor-Modus arbeiten, entgehen unserer Überprüfung.

Der SRP tritt in Aktion

Es stellt sich nun die Frage, wie man es in den Griff bekommen kann, einen Schreibschutz ausschließlich für den User-Modus zu realisieren. Hierzu bietet der 68030 zwei Möglichkeiten. Zum einen können die Deskriptortabellen so aufgebaut werden, daß eine Übersetzung in Abhängigkeit vom Zustand der Function Code Bits FC2-FC0 erfolgt [1]. (Diese Bits geben Aufschluß darüber, ob ein Befehls- oder Datenzugriff im User- oder Supervisor-Modus erfolgt.) Eine andere Lösung besteht darin, für User- und Supervisor-Zugriffe mit zwei unterschiedlichen Deskriptortabellen zu arbeiten. Genau dieses Verfahren wird vom Programm MEMWATCH eingesetzt.

Auf dem TT wird im Normalfall nicht unterschieden, ob auf eine Adresse im User- oder Supervisor-Modus zugegriffen wird. Daher kann sich TOS mit einer einzigen Deskriptortabelle begnügen, auf deren Beginn der CRP (CPU Root Pointer) zeigt. Die MMU stellt jedoch auch einen SRP (Supervisor Root Pointer) zur Verfügung. Soll im Supervisor-Modus eine andere Deskriptortabelle als im User-Betrieb verwendet werden, so zeigt der SRP auf den Start eben dieser Tabelle. Um dem Prozessor zu signalisieren, daß der Inhalt des SRP gültig ist, muß im TC-Register (Translation Control) lediglich des SRE-Bit (Supervisor Root Enable) gesetzt werden.

Da das Betriebssystem in der Lage sein soll, ohne Einschränkungen auf den Hauptspeicher zuzugreifen, ist es sinnvoll, an den Übersetzungsvorschriften für den Supervisor-Modus nichts zu ändern. Der SRP kann also (wie sonst der CRP) auf die standardmäßige Deskriptortabelle zeigen. Lediglich für den CRP muß eine neue Tabelle angelegt werden. In dieser wird dann von MEMWATCH je nach Bedarf vermerkt, ob eine Speicherseite gegen Schreibzugriffe geschützt ist.

Im Mittelpunkt steht MSHRINK

Eine Routine, die jedes auf dem TT gestartete Programm überwachen soll, muß sich in den GEMDOS-Vektor einklinken und in Aktion treten, sobald ein neues Programm gestartet wird. Diesen Zeitpunkt erkennt man am sichersten daran, daß hierzu die PEXEC-Routine des GEMDOS aufgerufen wird. Es liegt also nahe, auf einen *PEXEC*-Aufruf zu warten, um anschließend die MMU so zu konfigurieren, daß Schreibzugriffe auf den Code des gestarteten Programms verhindert werden. Das hört sich zwar einfach an, ist es aber leider nicht. Wird PEXEC aufgerufen, befindet sich das betreffende Programm noch gar nicht im Speicher. Man hat somit keine Informationen darüber, in welchem Bereich des Hauptspeichers das TEXT-Segment zu liegen kommen wird. Es ist zwar durchaus realisierbar, daß ein residentes Programm dafür sorgt, andere Programme zu laden und zu starten. Hierzu stellt das GEMDOS diverse PEXEC-Modi zur Verfügung. Dummerweise müßte man sich jedoch darum kümmern, das Programm zu relozieren, bei Bedarf den Speicher zu löschen und einen Teil der Basepage mit

gültigen Daten zu füllen. Außerdem wären noch die Bits im Programm-Header zu analysieren, die ausschlaggebend dafür sind, ob ein Programm im ST- oder TT-RAM ausgeführt werden soll. Um diesen Aufwand zu vermeiden, greift MEMWATCH zu einer alternativen Methode.

Wenn man davon ausgeht, daß selbst unsauber geschriebene Programme sich an gewisse Richtlinien halten müssen, um überhaupt korrekt zu laufen, liegt es nahe, sich nicht in PEXEC, sondern in *MSHRINK* einzuklinken. Diese Routine sollte von jedem Programm kurz nach dem Programmstart aufgerufen werden, um dem Betriebssysten den nicht benötigen Speicher zurückzugeben. Zum Zeitpunkt eines *MSHRINK*-Aufrufs befinden sich bereits alle für einen Schreibschutz des TEXT-Segments relevanten Daten im Speicher. Anhand dieser Angaben kann man Beginn und Länge des Programmcodes ermitteln. MEMWATCH besorgt sich hierzu den Pointer auf die Basepage des aktuellen Prozesses. Hierzu existiert seit TOS 1.02 („Blittertos") die Systemvariable *_run*, deren Adresse über die *_sysbase*-Struktur erhalten werden kann und die einen Zeiger auf die Basepage der aktuellen Applikation darstellt. Die Basepage wiederum enthält Informationen über Start und Länge der einzelnen Programmsegmente.

Beschränkung auf das TEXT-Segment

Per MMU ist der Adreßraum beim TT in Speicherseiten mit einer Größe von 32 KByte aufgeteilt. Dieser Wert stellt somit die kleinste Einheit dar, die schreibgeschützt werden kann. Nun ist es natürlich nicht so, daß ein TEXT-Segment genau an einer Seitengrenze liegt und einen Umfang von 32 KByte hat. Ein Schreibschutz per MMU wird sich also in der Regel auch auf Bereiche vor und hinter dem Programmcode erstrecken. Hinter dem TEXT-Segment befindet sich beispielsweise das DATA-Segment, für das MEMWATCH keinen Schreibschutz einrichten soll. Tritt nun ein Busfehler auf, darf nicht gleich davon ausgegangen werden, daß das TEXT-Segment modifiziert wurde. Zunächst muß geprüft werden, ob die Adresse, auf die zugegriffen wurde, tatsächlich im Programmcode der aktiven Applikation liegt. Hierzu bedient sich MEMWATCH erneut der Basepage. Ein Vergleich der Zugriffsadresse mit Start und Länge des TEXT-Segments sorgt für Klarheit. Nur dann, wenn tatsächlich versucht wurde, auf eine Adresse innerhalb des Programms zu schreiben, wird ein Warnton ausgegeben. Andernfalls wird der Zugriff lediglich im Supervisor-Modus wiederholt. Was die Geschwindigkeit bei der Programmausführung betrifft, leidet diese je nach Lage des TEXT-Segments selbstverständlich darunter, daß auch Zugriffe auf benachbarte Speicherbereiche einen Busfehler auslösen. Die Exception-Behandlung braucht schließlich ihre Zeit, auch wenn ein Schreibzugriff erlaubt ist.

Das Ende vom Lied

Wie genau MEMWATCH vorgeht, um den Schreibschutz zu installieren, kann man dem kommentierten Assembler-Quelltext entnehmen. Es darf natürlich nicht vergessen werden, daß nach dem Beenden eines Programms die schreibgeschützten Speicherseiten wieder für Schreibzugriffe aus dem User-Modus zugänglich sein müssen. Um dies zu erreichen, genügt es, Aufrufe von *PTERM0* und *PTERM* zu überwachen. Eine dieser GEMDOS-Funktionen muß jedes Programm aufrufen, um sich ordnungsgemäß zu beenden. Somit bietet es sich an, während eines Aufrufs obiger Routinen die nötigen MMU-Manipulationen vorzunehmen, um den Schreibschutz zu entfernen. Programme, die sich resident im Speicher verankern, beenden sich mittels *PTERMRES*. Hier greift MEMWATCH nicht ein, so daß die TEXT-Segmente residenter Routinen weiterhin gegen Überschreiben geschützt bleiben.

Die Bedeutung des VBR

Alle Programme, die sich wie MEMWATCH in ihrer Funktion darauf stützen, daß der Busfehler-Vektor zur Realisierung von MMU-Manipulationen geändert wird, haben mit einem nicht unerheblichen Problem zu kämpfen. Es darf auf keinen Fall vorkommen, daß ein anderes Programm, das nicht mit der Programmierung der MMU vertraut ist, diesen Vektor in seinem Sinne umbiegt. So gibt es einige Programme, die beim Auftreten einer Exception versuchen, diese eigenständig zu behandeln. Solche Programme geben dann nicht einfach zwei Bomben auf dem Bildschirm aus, wie es TOS tut, sondern warten mit einer Fehlermeldung auf. Nun wäre es fatal, wenn MEMWATCH beim Auftreten eines Busfehlers nicht mehr dazu käme, die Ursache für eine Busfehler-Exception zu überprüfen. Wurde diese nämlich durch einen Schreibzugriff auf das TEXT-Segment einer Applikation verursacht, muß sichergestellt sein, daß das Hauptprogramm nach der Ausgabe des Warntons weitergeführt wird. MEMWATCH muß aus diesem Grund also unbedingt das erste Programm sein, das nach einem Busfehler aufgerufen wird.

Dies läßt sich dadurch sicherstellen, daß man das VBR (Vector-Base-Register) in geeigneter Weise einsetzt. Dieses Register existiert bei den Prozessoren der 68000-Familie erst seit dem 68010. Das VBR definiert, an welcher Adresse sich die Tabelle mit den Exception-Vektoren befindet. Beim 68000-Prozessor liegen diese Vektoren stets am Beginn des Adreßraums, also ab Adresse 0. Alle Programme, die nicht an neuere Prozessoren angepaßt sind (also auch nicht in der Lage sind, das VBR selber zu manipulieren), gehen davon aus, daß die Exception-Vektoren ab Adresse 0 angelegt sind. Das TOS des TT sieht die Situation ähnlich. (Dies zeigt eine unzureichende Anpassung des TOS an den 68030.) Werden Vektoren über den *SETEXEC*-Aufruf des BIOS geändert, spricht TOS unabhängig vom Inhalt des VBR stets Adressen am Beginn des Hauptspeichers an. Obwohl dieses Verhalten von TOS nicht unbedingt korrekt ist, ist es für unseren Zweck überaus praktisch.

Hinters Licht geführt

MEMWATCH richtet nun die Vektortabelle anderswo im Speicher ein. Programme, die die Exception-Vektoren an der üblichen Stelle verändern, bewirken damit überhaupt nichts mehr. Die alten Vektoren werden schließlich gar nicht mehr verwendet, so daß deren Inhalt zunächst keine Rolle spielt. Tritt nun ein Busfehler auf, wird über den neuen Busfehler-Vektor stets die Exception-Behandlung von MEMWATCH aufgerufen. Auch der GEMDOS-Vektor wird auf diese Weise verlegt, so daß bei GEMDOS-Aufrufen ebenfalls zuerst in MEMWATCH eingesprungen wird. Da für das Programm lediglich zwei Vektoren von Bedeutung sind, genügt es, die Exception-Behandlung so einzurichten, daß beim Auftreten einer Exception, die nicht von MEMWATCH verwaltet wird, die alten Vektoren ab Adresse 0 angesprungen werden. Dies läßt sich mit minimalem Aufwand dadurch verwirklichen, daß man den Vektor-Offset, den der 68030 bei jeder Exception auf dem Stack ablegt, entsprechend auswertet. Mit einer einzigen Routine ist es so möglich, für alle Vektoren eine einheitliche Exception-Behandlung zu gewährleisten.

Geschwindigkeitsgewinn

Bleiben wir noch ein wenig beim VBR. Läuft MEMWATCH im TT-RAM, liegt

selbstverständlich auch die neu eingerichtete Vektortabelle im schnellen RAM. Da es dem Prozessor so möglich ist, schneller auf die Exception-Vektoren zuzugreifen, als wenn diese im ST-RAM lägen, kann man durch geeignete Manipulationen des VBR eine leichte Beschleunigung des TT erzielen. Bei MEMWATCH kommt dieser Effekt natürlich kaum zum Tragen, da hier innerhalb der Exception-Bearbeitung eine Neuberechnung der Vektoradressen erfolgt. Bei anderen Anwendungen kann das aber durchaus anders aussehen. Wird die Tabelle der Exception-Vektoren per VBR verschoben, muß die neue Adresse unbedingt durch 16 teilbar sein. Die Vektortabelle muß also stets auf einer sogenannten *Cacheline* beginnen.

Unabhängig vom VBR hat es mit solchen Cachelines eine besondere Bewandtnis. Routinen, die auf einer Cacheline beginnen, werden vom 68030 schneller ausgeführt, als wenn sie sich an einer Adresse befinden, die sich nicht durch 16 teilen läßt. Dieser Effekt hängt damit zusammen, daß es der Burst-Modus des 68030 erlaubt, eine komplette Cacheline in kürzester Zeit in den internen Prozessor-Cache zu übertragen. Arbeitet man mit einem Assembler, der 68020/68030-Code erzeugen kann, steht in der Regel ein Befehl zur Verfügung, der ein „Alignment" einer Routine erlaubt. Beim MAS handelt es sich um die *align*-Direktive. So besagt *align 16*, daß der nächste Befehl so assembliert wird, daß er auf einer Cacheline zu liegen kommt.

Ataris Versäumnis

Es läßt sich zeigen, daß Programme, die häufig benötigte Routinen auf einer Cacheline plazieren, um bis zu 8% schneller ablaufen, als wenn diese Routinen lediglich auf einer Wortgrenze liegen. Es empfiehlt sich also, insbesondere Unterprogramme, die periodisch (z.B. während des VBL) aufgerufen werden, auf eine Cacheline zu legen. Nun hört sich das zunächst ganz einfach an, denn man kann dazu die entsprechenden align-Befehle einsetzen. aber die Sache hat leider einen Haken, der mit der Struktur der bisherigen TOS-Versionen zusammenhängt. Zwar kann ein Assembler dafür sorgen, daß ein Befehl relativ zum Programmbeginn an einer ganz bestimmten Adresse zu liegen kommt, aber damit das Alignment auch wirklich funktioniert, muß der Beginn eines Programms unbedingt auf einer Cacheline liegen. TOS gewährleistet dies jedoch nicht. Ein Programm wird beim Start mittels *PEXEC* lediglich an eine gerade Adresse geladen. Ob es sich dabei um eine Cacheline handelt, bleibt dem Zufall überlassen. Meist ist dies nicht der Fall. Somit lassen sich Befehle zum Alignment von Programmcode bisher nicht sinnvoll einsetzen. Dennoch sollte man diese Möglichkeit in eigenen Programmen berücksichtigen. Sollte es einmal eine TOS-Version geben, die den Beginn eines Programms auf eine Cacheline plaziert, so profitiert man von der erhöhten Ausführungsgeschwindigkeit. Man kann nur hoffen, daß Atari diese Problematik überdenkt und im Sinne des Anwenders handelt.

Übeltäter en masse

Nach diesem kurzen Exkurs nun aber zurück zu MEMWATCH. Sie sind jetzt in der Lage, sich ein Bild darüber zu machen, wie sauber die von Ihnen genutzte Software programmiert ist. Im großen und ganzen zeigt sich, daß eine ganze Reihe an Programmen für ST und TT mit selbstmodifizierendem Code arbeitet. In den meisten Fällen zeigt MEMWATCH eine Codemanipulation jedoch nur kurz nach einem Programmstart an. Hier modifiziert sich ein Programm lediglich in seiner Initialisierungsphase. Es gibt jedoch auch Programme, die ihren Code ständig manipulieren. Dabei handelt es sich insbesondere um Software, die bereits an anderer Stelle [1] unangenehm aufgefallen ist. Zu wundern braucht man sich über diese Feststellung wohl nicht.

XBRA-Dilemma

Nun mag sich der eine oder andere Programmierer gar nicht darüber im klaren sein, daß er in seinen Programmen selbstmodifizierenden Code verwendet. So verleitet beispielsweise der Einsatz des XBRA-Verfahrens [2] dazu, entsprechende Routinen zu verwenden. Häufig wird die XBRA-Struktur direkt vor dem Einsprungpunkt für den umgebogenen Vektor errichtet und befindet sich somit im TEXT-Segment eines Programms. Das muß jedoch nicht zwangsweise so sein.

Zur Verdeutlichung noch einmal kurz die Grundlagen des XBRA-Systems. Residente Programme, die einen Systemvektor verbiegen, überschreiben diesen Vektor nicht einfach mit einem neuen Inhalt, sondern merken sich den alten Vektor anhand einer XBRA-Struktur, die den folgenden Aufbau hat:

```
typedef struct
{
  char xb_magic[4];      /* XBRA-Kennung
                            "XBRA" */
  char xb_id[4];         /* ID des resden
                            ten Programms */
  long xb_oldvec;        /* ursprünglicher
                            Vektorinhalt */
```

Das XBRA-Protokoll ermöglicht es residenten Programmen unter anderem, sich zu deinstallieren, ohne dabei andere Programme, die die gleichen Vektoren verbiegen, in ihrer Funktion einzuschränken. Darüber hinaus kann ein Programm anhand seiner eigenen XBRA-Struktur erkennen, ob es bereits installiert ist. Mehrfachinstallationen lassen sich so leicht verhindern.

Der XBRA-Datenblock muß sich direkt vor der Routine befinden, auf die der umgebogene Vektor zeigt. Da sich der Programmcode im TEXT-Segment befindet, hat dies zur Folge, daß sich die XBRA-Struktur oft ebenfalls in diesem Segment wiederfindet. Bei der Programminitialisierung wird hier dann der alte Vektorinhalt eingetragen - und schon ist es passiert: das TEXT-Segment wurde modifiziert.

XBRA im DATA-Segment

Damit es beim Einrichten der XBRA-Struktur nicht notwendig ist, das TEXT-Segment eines Programms zu beeinflussen, muß diese Strukur anderswo angelegt werden, beispielsweise im DATA-Segment. Schauen wir uns an, wie man hierzu vorgehen kann.

Da der Programmcode, den der verbogene Vektor anspringt, direkt hinter der XBRA-Strukur liegen muß, bietet es sich an, an dieser Stelle keinen kompletten Programmteil, sondern lediglich einen Sprungbefehl auf die eigentliche Routine abzulegen. Dies würde einer erweiterten Struktur entsprechen, die im DATA-Segment eingerichtet werden kann:

```
typedef struct
{
  char xb_magic[4];   /* XBRA-Kennung
                         "XBRA" */
  char xb_id[4];      /* ID des residen
                         ten Programms */
  long xb_oldvec;     /* ursprünglicher
                         Vektorinhalt */
  int  xb_jmp;        /* $4EF9 als JMP-
                         Opcode */
  long xb_newvec;     /* neuer Vektor */
}
```

Bei der Initialisierung dieser Daten muß ein Programm lediglich das Langwort *xb_newvec* durch die Adresse jener Routine ersetzen, die über den geänderten Vektor angesprungen werden soll. Diese Routine befindet sich logischerweise im TEXT-Segment.

Wie man sieht, läßt sich mit wenig Aufwand dafür sorgen, daß die Verwendung des XBRA-Protokolls nicht in selbstmodifizierendem Code endet. Auch in anderen Fällen kann man mit durchdachten Routinen auf solchen Code verzichten und so dem Ziel der sauberen Programmierung ein weiteres Stück näherkommen.

PMMU-Handler

Abschließend möchte ich noch auf ein System hinweisen, das Kollisionen vermeiden soll, falls mehrere Programme gestartet werden, die die 68030-MMU nutzen. In den allermeisten Fällen vertragen sich solche Programme nicht miteinander, da sich die MMU nicht ohne weiteres für mehrere Aufgaben gleichzeitig einsetzen läßt. Aus diesem Grund wird eine möglichst einheitliche Methode benötigt, die eine Überprüfung der MMU-Aktivität erlaubt. Einige Programme für den TT nutzen den cookie-jar [4], um bei dieser Problematik Abhilfe zu schaffen.

Programme, die die MMU programmieren, sollten einen cookie mit der Kennung *PMMU* einrichten. Ist der zugehörige cookie-Parameter kein Null-Pointer, zeigt er auf die Adresse eines Handlers, der über definierte Aufrufe gewisse MMU-Manipulationen erlaubt. In der nächsten Ausgabe der ST-Computer wird diese Thematik voraussichtlich ausführlich behandelt. An dieser Stelle soll lediglich der Hinweis stehen, daß die bloße Anwesenheit des *PMMU*-cookies signalisiert, daß die MMU bereits genutzt wird. So haben Programme die Möglichkeit, sich nicht zu installieren, falls dieser cookie vorhanden ist. Selbstverständlich kann ein Programm neben dem MMU-cookie einen weiteren, programmspezifischen cookie einrichten. Schließlich ist es nicht untersagt, mehr als einen cookie pro Programm zu verwenden. Es existiert bereits Software, die den *PMMU*-cookie einrichtet bzw. auswertet. Dazu zählen neben MEMWATCH beispielsweise die virtuelle Speicherverwaltung OUTSIDE ab Version 1.03, ROMSPEED V1.2, ROMRAM V1.1J, VRAM sowie neue SYS_MON-Versionen.

US

Literatur:

[1] „Screenwatch - Direkter Bildschirmzugriff - nein danke", ST-Computer 7,8/91

[2] Jankowski, Reschke, Rabich „ATARI ST Profibuch", SYBEX-Verlag

[3] Steve Williams, „68030 Assembly Language Reference", Addison-Wesley Publishing Inc.

[4] Rolf Kotzian, „Das cookie-Jar-Prinzip", ST-Computer 12/90

```
****************************
*                          *
* MEMWATCH V1.0            *
*                          *
* Schreibschutz für das    *
* TEXT-Segment auf dem TT  *
*                          *
* (c) 1991 MAXON Computer  *
* by Uwe Seimet            *
*                          *
****************************


GEMDOS   = 1
CCONWS   = 9
SUPER    = 32
PTERMRES = 49
MSHRINK  = 74
PTERM    = 76


BIOS     = 13
BCONOUT  = 3


_sysbase = $4f2
_p_cookies = $5a0


         text

         move.l 4(sp),a6  ;Pointer auf
                          ;Basepage
         lea stack+400,sp
         move.l 12(a6),a1 ;Länge des
                          ;TEXT-Segments
         add.l 20(a6),a1  ;DATA-Segment
         add.l 28(a6),a1  ;BSS-Segment
         lea $100(a1),a1  ;für Basepage
         move.l a1,prglen ;Programmlänge
                          ;merken
         pea (a1)
         pea (a6)
         clr -(sp)
         move #MSHRINK,-(sp)
         trap #GEMDOS     ;Restspeicher
                          ;freigeben
         lea 12(sp),sp
         tst.l d0         ;alles klar?
         bne quit         ;leider nicht-

         clr.l -(sp)
         move #SUPER,-(sp) ;Supervisor-
         trap #GEMDOS     ;Modus
         addq.l #6,sp
         move.l d0,d7     ;SSP merken

         lea sterr,a5
         move.l _p_cookies,d0 ;kein
         beq error            ;cookie jar-
         move.l d0,a0
testjar: movem.l (a0)+,d0-d1
         tst.l d0
         beq endjar
         cmp.l #"_MCH",d0  ;Computertyp?
         bne nomch         ;nein-
         swap d1
         subq #2,d1        ;TT?
         bne error         ;nein-
nomch:   cmp.l #"PMMU",d0  ;MMU-Programm
                           ;aktiv?
         bne testjar       ;nein-
         lea mmuerr,a5
         bra error

endjar:  move.l #"PMMU",-8(a0)   ;kein
         clr.l -4(a0)        ;PMMU-Handler
         movem.l d0-d1,(a0)

         move.l #table+15,d6 ;Deskriptor-
                             ;Tabelle
         and #$fff0,d6       ;auf Cacheline
                             ;ausrichten
         pmove crp,crpreg    ;alter CRP
         pmove crpreg,srp
         move.l crpreg+4,a1 ;Pointer auf
                            ;alte Tabelle
         move.l a1,a2
         move.l d6,crpreg+4 ;für neuen
                            ;CRP

         move.l d6,a0
         moveq #63,d0
tcopy:   move.l (a1)+,d1
         move.l d1,d2
         and #$02,d2       ;Tabellen-
         cmp #$02,d2       ;Deskriptor?
         bne notable       ;nein-
         sub.l a2,d1
         add.l d6,d1
notable: move.l d1,(a0)+   ;Tabelle
         dbra d0,tcopy     ;kopieren

         move.l a0,d1
         or #2,d1
         moveq #13,d0
         move.l d6,a1
         lea $c0(a1),a1
cdes:    move.l d1,(a1)+   ;Deskriptoren
         add.l #128,d1     ;für ST-RAM
         dbra d0,cdes

         moveq #1,d1
         move #511-64,d0   ;Seiten-
```

```
dloop0:  move.l d1,(a0)+     ;Deskriptoren
         add.l #$8000,d1     ;für ST-RAM
         dbra d0,dloop0

         move.l a0,d2
         move.l a0,d1
         add.l #64,d1
         or #2,d1
         moveq #15,d0
dloop1:  move.l d1,(a0)+
         add.l #128,d1
         dbra d0,dloop1

         move.l #$01000001,d1
         move #511,d0        ;Seiten-
dloop2:  move.l d1,(a0)+     ;Deskriptoren
         add.l #$8000,d1     ;für TT-RAM
         dbra d0,dloop2
         or #2,d2

         move.l d2,($44,d6.l)

         move.l #vectors+15,d0 ;VBR auf
         and #$fff0,d0       ;Cacheline
         move.l d0,a1        ;ausrichten
         move.l d0,a2
         movec vbr,a0
         move.l $08(a0),o_bus
         move.l $84(a0),o_gemdos
         move.w #255,d0
copyvec: move.l #newvec,(a1)+
         dbra d0,copyvec
         move.l #buserr,$08
         move.l #gemdos,$84
         movec a2,vbr

         lea dummy,a0        ;keine
*        pmove (a0),tt0      ;transparente
                             ;Übersetzung
         dc.l $f0100800      ;PMOVE (A0),TT0
         pmove crpreg,crp    ;neuer
                             ;Rootpointer
         pmove tc,tcreg
         bset #1,tcreg       ;SRE setzen
         pmove tcreg,tc

         move.l d7,-(sp)
         move #SUPER,-(sp)   ;Rückkehr in
         trap #GEMDOS        ;User-Modus
         addq.l #6,sp

         pea message
         move #CCONWS,-(sp)
         trap #GEMDOS
         addq.l #6,sp
         clr -(sp)
         move.l prglen,-(sp)
         move #PTERMRES,-(sp) ;zurück
         trap #GEMDOS          ;zum TOS


gemdos:
         lea 8(sp),a0        ;Supervisor-
         btst #5,(sp)        ;Modus?
         beq user            ;nein-
cont:    jmp ([o_gemdos])
user:    move.l usp,a0       ;sonst
                             ;User-Stack
         tst (a0)
         beq term            ;PTERM0-
         cmp #PTERM,(a0)
         beq term            ;PTERM-
         cmp #MSHRINK,(a0)
         bne cont
         move.l ([_sysbase],40),a0
         move.l (a0),a0      ;zeigt auf PD
         move.l 12(a0),d0
         ptestw #2,([a0],8),#7,a0
wrset:   bset #2,3(a0)       ;W-Bit
         addq.l #4,a0        ;setzen
         sub.l #$8000,d0
         bcc wrset           ;nächste Page-
         pflusha
         bra cont

term:
         move.l ([_sysbase],40),a0
         move.l (a0),a0      ;zeigt auf PD
         move.l 12(a0),d0
         ptestw #2,([a0],8),#7,a0
wrclr:   bclr #2,3(a0)       ;Schreibschutz
         addq.l #4,a0        ;entfernen
         sub.l #$8000,d0
         bcc wrclr           ;nächste Page-
         pflusha
         bra cont


error:
         pea (a5)            ;Fehlermeldung
         move #CCONWS,-(sp)  ;ausgeben
         trap #GEMDOS
         addq.l #6,sp
         pea inserr          ;nicht
         move #CCONWS,-(sp)  ;installiert
         trap #GEMDOS
         addq.l #6,sp
exit:    move.l d7,-(sp)
         move #SUPER,-(sp)   ;zurück in
         trap #GEMDOS        ;User-Modus
         addq.l #6,sp
quit:    clr -(sp)
         trap #GEMDOS


         align 16

newvec:
         move.l 4(sp),-(sp)  ;Vektoroffset
         and.l #$fff,(sp)    ;isolieren
         move.l ([sp]),(sp)
         rts


buserr:
         movem.l d0/a6,-(sp)
         lea 8(sp),a6        ;Throwaway-
         cmp #$1008,6(a6)    ;Stackframe?
         bne is_isp          ;nein-
         movec msp,a6
is_isp:  move.b 11(a6),d0
         movec d0,dfc        ;Deskriptor
         ptestw dfc,([16,a6]),#7 ;prüfen
         subq.l #2,sp
         pmove psr,(sp)
         btst #3,(sp)+       ;W-Bit testen
         bne wrterr
         movem.l (sp)+,d0/a6
         move.l ([o_bus]),-(sp)
                             ;weiter im TOS
         rts

wrterr:
         move.l 16(a6),d0
         move.l ([_sysbase],40),a6
         move.l (a6),a6      ;zeigt auf PD
         sub.l 8(a6),d0      ;oberhalb oder
         bcs noerr           ;unterhalb des
         cmp.l 12(a6),d0     ;TEXT-
         bcc noerr           ;Segments?
         movem.l a0-a2/d0-d2,-(sp)
         move #7,-(sp)       ;BEL
         move #2,-(sp)
         move #BCONOUT,-(sp) ;Warnton
         trap #BIOS          ;ausgeben
         addq.l #6,sp
         movem.l (sp)+,a0-a2/d0-d2
noerr:   movem.l (sp)+,d0/a6
         bset #2,11(sp)      ;Befehl im
         rte                 ;Supervisor-
                             ;Modus
                             ;wiederholen


         data


message: dc.b $0d,$0a,"MEMWATCH V1.0 "
         dc.b "installiert",$0d,$0a
         dc.b "(c) 1991 by Uwe Seimet",$0d,
              $0a,0


*Diverse Fehlermeldungen
```

```
sterr:   dc.b $0d,$0a,"MEMWATCH läuft nur "
         dc.b "auf einem TT!",0

mmuerr:  dc.b $0d,$0a,"Es ist bereits ein "
    dc.b "MMU-Program aktiv!",0

inserr:  dc.b $0d,$0a,"MEMWATCH V1.0 "
         dc.b "nicht installiert",$0d,$0a,0


         bss

stack:   ds.l 100

crpreg:  ds.l 2

tcreg:    ds.l 1

prglen:   ds.l 1

dummy:    ds.l 1

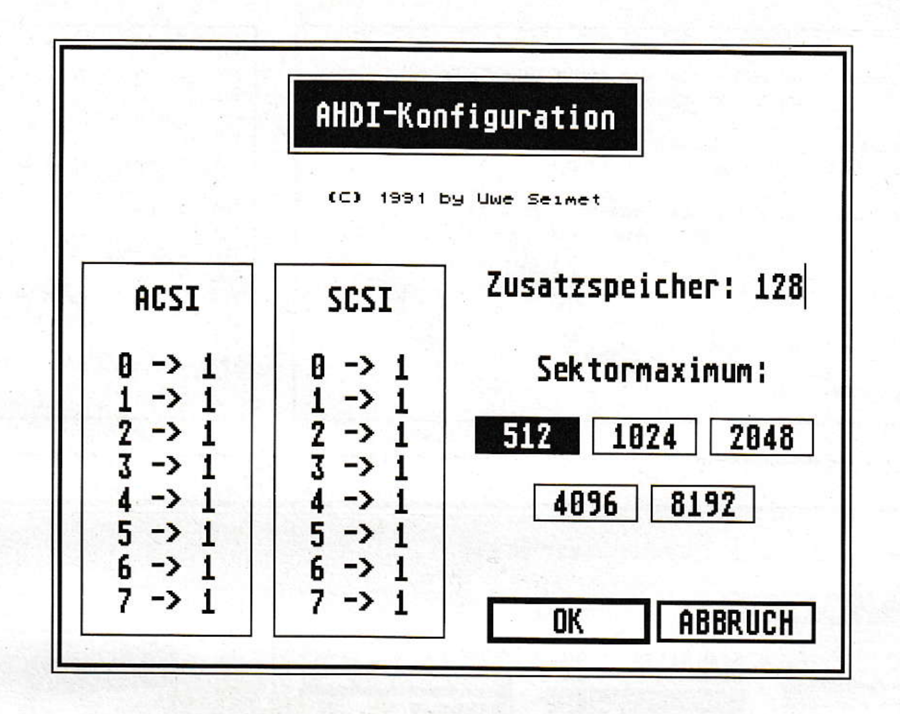
o_bus:    ds.l 1

o_gemdos:ds.l 1

table:    ds.b 512+1792+64+2048+15
          ;Deskriptortabelle

vectors: ds.b 1024+15    ;für Vektoren
```

AHDI paßt sich an



Bereits seit dem Spätsommer 1989 liegt der Atari-Festplattentreiber AHDI in der Version 3.0 vor, die inzwischen durch die Version 4.0, die in erster Linie für die Besitzer des Atari TT interessant ist, ersetzt wurde. Eine besondere Eigenschaft beider Treiber ist der Umstand, daß diese konfiguriert werden können. Leider gehört ein geeignetes Konfigurationsprogramm nicht zum Lieferumfang. Grund genug, selber ein solches Programm zu schreiben.

Konfigurierbare Festplattentreiber sind heute keine Seltenheit mehr. Die meisten Vertreiber von Festplatten für den Atari legen Wert darauf, ihre Platten mit Treiber-Software auszustatten, deren Funktionsumfang dem der Atari-Treiber in der Regel überlegen ist. Dennoch ist der AHDI-Treiber weit verbreitet, was nicht zuletzt daran liegt, daß die TTs und MegaSTEs standardmäßig mit diesem Treiber ausgeliefert werden.

Nun scheint AHDI auf den ersten Blick über keinerlei Möglichkeiten zu verfügen, individuelle Anpassungen an eine bestimmte Arbeitsumgebung vorzunehmen. Ganz so ist es aber nun doch nicht. Ein Blick in [1] zeigt, daß AHDI durchaus gewisse Konfigurationsmöglichkeiten bietet.

Patch-Bereiche

Innerhalb des AHDI-Programmcodes existiert ein für den Benutzer legal zugänglicher Datenbereich, der diverse Parameter enthält. Es handelt sich um eine Struktur, deren Aufbau Sie aus Tabelle 1 entnehmen können.

Was besagen diese Variablen (für die es keine offiziellen Bezeichnungen zu geben scheint) nun im einzelnen? Für TOS-Versionen, die älter als das Blitter-TOS (TOS 1.02) sind, spezifiziert *chunk*, um wieviele Speicherblöcke der interne Speicher des GEMDOS erweitert werden soll. Diese Erweiterung ist beim TOS 1.00 für einen fehlerfreien Festplattenbetrieb notwendig. Beim Blitter-TOS und neueren TOS-Versionen wird die Angabe in *chunk* von AHDI nicht mehr ausgewertet.

defbigse legt die maximale logische Sektorgröße fest, für die im Verlauf der Treiberinitialisierung Speicherplatz bereitgehalten wird. AHDI ermittelt diesen Wert beim Programmstart. Partitionen mit größeren Sektoren können vom Treiber anschließend nicht verwaltet werden. Um AHDI mitzuteilen, daß die Sektorgröße einer Partition die beim Booten vorgefundenen Werte überschreiten kann, muß in *defbigse* eine maximale Sektorgröße angegeben werden.

Arbeitet man ausschließlich mit Festplatten, so ist *defbigse* nicht weiter von Bedeutung. Schließlich kann es in diesem Fall nicht vorkommen, daß nachträglich eine Partition ins System integriert werden soll, die beim Booten noch nicht vorhanden war.

Anders sieht es dagegen aus, wenn man im Besitz einer Wechselplatte ist. Wird hier die Cartridge gewechselt, so ist es durchaus denkbar, daß sich auf dem neu eingelegten Medium eine Partition befindet, die größere Sektoren besitzt als alle bisher vorhandenen Partitionen. Der Inhalt von defbigse muß diesen Maximalwert widerspiegeln, damit alle Partitionen der neuen Cartridge ins System eingebunden werden können. Ist der Wert von defbigse größer als die maximale Sektorgröße, so reservieren sich Cache-Programme für die Festplatte unter Umständen unnötig viel Speicherplatz.

AHDI kann übrigens Sektoren mit einer logischen Sektorgröße von maximal 8192 Bytes verwalten. Die Sektorgröße ist in der Regel von der Kapazität einer Partition abhängig. Es gilt folgende Zuordnung:

Kapazität	Sektorgröße
< 16 MByte	512 Bytes
< 32 MByte	1024 Bytes
< 64 MByte	2048 Bytes
< 128 MByte	4096 Bytes
<= 256 MByte	8192 Bytes

Wie man sieht, kann AHDI Partitionen mit bis zu 256 Megabyte (= 1/4 Gigabyte) Kapazität verwalten. Bleibt noch anzumerken, daß es sich bei den obigen Werten um logische Angaben handelt. In Wirklichkeit arbeiten Festplatten für den ST oder TT stets mit 512 Bytes pro Sektor. Bei großen Partitionen werden mehrere dieser physikalischen Sektoren zu einem logischen Sektor zusammengefaßt.

Wechselhaft

Auch die restlichen Variablen des AHDI-Treibers sind in erster Linie für Besitzer von Wechselplatten interessant. Nicht nur die Sektorgrößen können auf verschiedenen Cartridges variieren, auch die Zahl der Partitionen pro Cartridge kann unterschiedlich sein. Liegt beim Start von AHDI ein Medium mit nur einer Partition im Laufwerk, und legt man anschließend eine Platte mit zwei Partitionen ein, so darf man sich nicht wundern, wenn von diesen beiden Partitionen nur die erste angesprochen werden kann.

Die Felder *acsi_def_ndrv* und *scsi_def_-ndrv* erlauben es, für jedes angeschlossene Wechselplattenlaufwerk festzulegen, für wieviele Partitionen Laufwerkskennungen (also C: bis P:) reserviert werden sollen. Hier muß sinnvollerweise ein Wert angegeben werden, der der maximalen Zahl der Partitionen entspricht, die auf einer Cartridge vorkommen können. Der Standardwert liegt für alle Geräte bei 1. Diese Werte werden vom Treiber nur dann verwendet, wenn es sich bei einem Laufwerk um eine Wechselplatte handelt.

Was *scsi_def_ndrv* betrifft, so existieren über dieses Feld keine offiziellen Angaben von Atari. Betrachtet man sich den Variablenbereich von AHDI 4.0, stellt man jedoch fest, daß die für AHDI 3.0 beschriebene Anordnung von *acsi_devs* und *acsi_def_ndrv* in analoger Weise auch für Platten am SCSI-Bus des TT eingeführt wurde.

```
struct
{
     unsigned int magic;              /* $F0AD, falls konfigurierbar */
     unsigned int version;            /* Treiberversion, z.B. $0300 */
     int chunk;                       /* zusätzliche Ordner */
     int defbigse;                    /* maximale Sektorgröße */
     int acsi_devs;                   /* Zahl der ACSI-Platten (8) */
     char acsi_def_ndrv[acsi_devs];   /* reservierte ACSI-Partitionen */
     int scsi_devs;                   /* Zahl der SCSI-Platten (8) */
     char scsi_def_ndrv[scsi_devs];   /* reservierte SCSI-Partitionen */
} ahdi_vars;
```

Tabelle 1

Konfiguration per Programm

Alle für AHDI beschriebenen Konfigurationsdaten können mit dem abgedruckten Programm AHDICONF den eigenen Bedürfnissen entsprechend eingestellt werden. Hierzu muß zunächst die zu konfigurierende Treiberdatei (AHDI.PRG bzw. SHDRIVER.SYS) im Fileselector aufgesucht werden. Handelt es sich hierbei tatsächlich um AHDI 3.0 oder 4.0, so erscheint anschließend eine Dialogbox, die eine Neukonfigurierung erlaubt (Bild).

Für jedes Laufwerk kann die Zahl der zu reservierenden Partitionen getrennt angegeben werden. Handelt es sich bei der Treiberdatei um AHDI 3.0, lassen sich nur Angaben zu den ACSI-Platten machen, da der SCSI-Bus erst ab AHDI 4.0 unterstützt wird. Nachdem die Konfigurierung erfolgreich abgeschlossen wurde, muß der Treiber neu gestartet bzw. das System neu gebootet werden, damit die geänderten Angaben wirksam werden.

Beim Compilieren des Programms muß darauf geachtet werden, daß der C-Compiler Strings gleichen Inhalts nicht zu einem einzigen String zusammenlegt. Eventuell muß dies über eine Compiler-Option gesteuert werden.

US

Literatur:
[1] „AHDI 3.00 Release Notes", Atari Corp.

```
/**************************/
/* AHDICONF               */
/*                        */
/* zur Konfiguration von  */
/*                        */
/* AHDI V3.0 oder V4.0    */
/* (c) 1991 MAXON Computer*/
/* by Uwe Seimet          */
/**************************/


#define EXTERN extern

#include "ahdiconf.rsh"
#include "ahdiconf.rh"
#include <vdi.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ext.h>


/* Offset des Parameterbereichs */
/* zum Beginn der AHDI-Programmdatei */
#define PARSTART 40

/* 42 Parameterbytes bei AHDI 4.0 */
#define PARLEN 42

#define TRUE 1
#define FALSE 0

typedef enum _bool boolean;


int contrl[11],
    intin[80],
    intout[45],
    ptsin[32],
    ptsout[32];

int work_in[12],
    work_out[57];

int g_handle;

int gl_hchar,
    gl_wchar;


char ahdiname[150];


/* Konfigurations-Variablen */
struct
    {
     unsigned int magic;
     unsigned int version;
     int          chunk;
     int          defbigse;
     int          acsi_devs;
     char         acsi_def_ndrv[8];
     int          scsi_devs;
     char         scsi_def_ndrv[8];
     } ahdi_vars;


/* Funktions-Prototypen */
boolean open_vwork(void);
boolean open_driver(void);
boolean close_driver(void);
void get_par(void);
void put_par(void);
boolean do_dialog(void);
void rsrc_init(void);



int main()
{
```

→

Listing 1: Der C-Quelltext zu AHDICONF

```
  if ((appl_init())!=-1)
    {
    if (open_vwork())
      {
      /* Treiberdatei öffnen */
      if (open_driver())
        {
        /* Parameter auswerten */
        get_par();
        /* Resource-Daten umrechnen */
        rsrc_init();
        /* Eingaben sind erwünscht */
        if (do_dialog())
          {
          /* Parameter auswerten */
          put_par();
          /* Treiber konfigurieren */
          close_driver();
          }
        }
      v_clsvwk(g_handle);
      }
    }
  appl_exit();
  return(0);
}


/* Workstation öffnen */
boolean open_vwork()
{
  int gl_wbox,gl_hbox;

  register int i;

  for(i=1; i<10; work_in[i++]=0);
  work_in[10]=2;
  g_handle=graf_handle(&gl_wchar,&gl_hchar,
                  &gl_wbox,&gl_hbox);
  work_in[0]=g_handle;
  v_opnvwk(work_in,&g_handle,work_out);
  if (!g_handle) return(FALSE);
  return(TRUE);
}


/* Dialog führen */
boolean do_dialog()
{
  int fo_cx,fo_cy,fo_cw,fo_ch;
  int index;
  int exit;

  /* SCSI-Parameter nur ab AHDI V4.0 */
  if (ahdi_vars.version<0x0400)
    {
    rs_trindex[DIALOG][SCSI].ob_flags
        |= HIDETREE;
    for (index=SCSINUM; index<SCSINUM+8;
        index++)
      rs_trindex[DIALOG][index].ob_flags
        &= ~EDITABLE;
    }

  form_center(rs_trindex[DIALOG],&fo_cx,
        &fo_cy,&fo_cw,&fo_ch);
  form_dial(FMD_START,fo_cx,fo_cy,fo_cw,
        fo_ch,fo_cx,fo_cy,fo_cw,fo_ch);
  objc_draw(rs_trindex[DIALOG],0,3,fo_cx,
        fo_cy,fo_cw,fo_ch);
  graf_mouse(ARROW,NULL);
  exit=form_do(rs_trindex[DIALOG],CHUNK);
  form_dial(FMD_FINISH,fo_cx,fo_cy,fo_cw,
        fo_ch,fo_cx,fo_cy,fo_cw,fo_ch);
  return(exit!=ABORT);
}


/* Parameter aus Treiber holen */
void get_par()
{
  int index;
  char *te_ptext;

  te_ptext=
     rs_trindex[DIALOG][CHUNK].ob_spec
     .tedinfo->te_ptext;

  /* Zahl der zusätzlichen Ordner */
  itoa(ahdi_vars.chunk,te_ptext,10);

  /* Maximale Sektorgröße */
  index=ahdi_vars.defbigse/512-1;
  rs_trindex[DIALOG][DEFBIGSE+
     index].ob_state
     |= SELECTED;

  /* Default-Partitionen für ACSI */
  for (index=0; index<ahdi_vars.acsi_devs;
      index++)
    {
    te_ptext=rs_trindex[DIALOG][ACSINUM+
        index].ob_spec.tedinfo-
        >te_ptext;
    te_ptext[0]=ahdi_vars.acsi_def_ndrv
          [index]+'0';
    }

  /* dto für SCSI */
  if (ahdi_vars.version>=0x0400)
    {
    for (index=0;
        index<ahdi_vars.scsi_devs;
        index++)
      {
      te_ptext=rs_trindex[DIALOG][SCSINUM+
          index].ob_spec.tedinfo
          ->te_ptext;
      te_ptext[0]=ahdi_vars.scsi_def_ndrv
            [index]+'0';
      }
    }
}


/* Neue Parameterdaten erzeugen */
void put_par()
{
  int index=0;
  char *te_ptext;

  while(!(rs_trindex[DIALOG][DEFBIGSE+
      index]
      .ob_state && SELECTED)) index++;
  ahdi_vars.defbigse=(index+1)*512;
  te_ptext=rs_trindex[DIALOG][CHUNK]
      .ob_spec.tedinfo->te_ptext;
  ahdi_vars.chunk=atoi(te_ptext);
  for (index=0; index<ahdi_vars.acsi_devs;
      index++)
    {
    te_ptext=rs_trindex[DIALOG][ACSINUM+
          index].ob_spec.tedinfo
          ->te_ptext;
    ahdi_vars.acsi_def_ndrv[index]=te_ptext
      [0]-'0';
    }

  if (ahdi_vars.version>=0x0400)
    {
    for (index=0;
        index<ahdi_vars.scsi_devs;
        index++)
      {
      te_ptext=rs_trindex[DIALOG][SCSINUM+
          index].ob_spec.tedinfo
          ->te_ptext;
      ahdi_vars.scsi_def_ndrv[index]=
      te_ptext
        [0]-'0';
      }
    }
}


/* Treiberdatei öffnen, Parameter lesen */
boolean open_driver()
{
  char *dummy;
  int handle;
  int button;
  char filename[]="SHDRIVER.SYS";

  getcwd(ahdiname,MAXPATH);
  fsel_input(ahdiname,filename,&button);
  if (!button) return(FALSE);

  /* Zugriffspfad zusammensetzen */
  dummy=strrchr(ahdiname,'\\');
  if (dummy==NULL) return(FALSE);
  graf_mouse(BUSYBEE,NULL);
  strcpy(dummy+1,filename);
```

→

```

   handle=open(ahdiname,O_RDONLY);
   if (handle<0) return(FALSE);
   lseek(handle,(size_t)(PARSTART),
         SEEK_SET);
   if (read(handle,&ahdi_vars,
      (size_t)(PARLEN))
      <=0)
     {
     close(handle);
     return(FALSE);
     }
   close(handle);

   /* Korrekte Treiberversion? */
   if (ahdi_vars.magic!=0xf0ad)
       return(FALSE);
   return(TRUE);
}


/* Parameter zurückschreiben */
boolean close_driver()
{
   int handle;
   int parlen=PARLEN;

   graf_mouse(BUSYBEE,NULL);

   /* AHDI 3.0 hat 10 Parameterbytes
      weniger */
   if (ahdi_vars.version<0x400) parlen-=10;

   handle=open(ahdiname,O_WRONLY);
   lseek(handle,(size_t)(PARSTART),
         SEEK_SET);
   if (write(handle,&ahdi_vars,
      (size_t)(parlen))
      !=parlen) return(FALSE);
   close(handle);
   form_alert(1,"[1][Der Treiber wurde|\
neu konfiguriert.][   OK   ]");
   return(TRUE);
}


/* Objektkoordinaten umrechnen */
void rsrc_init()
{
   register int i;

   for(i=0; i<NUM_OBS; i++)
       rsrc_obfix(rs_trindex
       [DIALOG],i);
}
```

Listing 2: Objektdaten für die Dialogbox

```
/* GEM Resource C Source */

#include <portab.h>
#include <aes.h>
#include "AHDICONF.H"

#if !defined(WHITEBAK)
#define WHITEBAK    0x0080
#endif
#if !defined(DRAW3D)
#define DRAW3D      0x0040
#endif

#define FLAGS9  0x0200
#define FLAGS10 0x0400
#define FLAGS11 0x0800
#define FLAGS12 0x1000
#define FLAGS13 0x2000
#define FLAGS14 0x4000
#define FLAGS15 0x8000
#define STATE8  0x0100
#define STATE9  0x0200
#define STATE10 0x0400
#define STATE11 0x0800
#define STATE12 0x1000
#define STATE13 0x2000
#define STATE14 0x4000
#define STATE15 0x8000

TEDINFO rs_tedinfo[] =
{ "(C) 1991 by Uwe Seimet",
  "",
  "",
  SMALL, 0, TE_LEFT , 0x1180, 0, -1, 23, 1,
  "1",
  "0 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "1 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "2 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "3 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "4 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "5 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "6 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "7 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "0 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "1 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "2 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "3 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "4 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "5 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "6 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "1",
  "7 -> _",
  "9",
  IBM  , 0, TE_LEFT , 0x1180, 0, -1, 2, 7,
  "000",
  "Zusatzspeicher: ___",
  "999",
  IBM  , 0, TE_CNTR , 0x1180, 0, -1, 4, 20
};

OBJECT rs_object[] =
{
  /******** Tree 0 DIALOG ****************/
  -1, 1, ABORT, G_BOX,          /* Object 0  */
  NONE, OUTLINED, (LONG)0x00021100L,
  0x0003, 0x0005, 0x002F, 0x0812,
  2, -1, -1, G_BUTTON,          /* Object 1  */
  NONE, SELECTED|OUTLINED, (LONG)"AHDI-
  Konfiguration",
  0x000E, 0x0001, 0x0414, 0x0002,
  3, -1, -1, G_TEXT,            /* Object 2  */
  NONE, NORMAL, (LONG)&rs_tedinfo[0],
  0x0010, 0x0004, 0x0410, 0x0001,
  SCSI, 4, 12, G_BOX,           /* Object 3  */
  NONE, NORMAL, (LONG)0x00FF1101L,
  0x0401, 0x0806, 0x000A, 0x000B,
  ACSINUM, -1, -1, G_STRING,
                                /* Object 4  */
```

→

```
NONE, NORMAL, (LONG)"ACSI",
0x0003, 0x0800, 0x0004, 0x0001,
6, -1, -1, G_FTEXT,/* Object 5 ACSINUM */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[1],
0x0002, 0x0802, 0x0006, 0x0001,
7, -1, -1, G_FTEXT,        /* Object 6  */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[2],
0x0002, 0x0803, 0x0006, 0x0001,
8, -1, -1, G_FTEXT,        /* Object 7  */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[3],
0x0002, 0x0804, 0x0006, 0x0001,
9, -1, -1, G_FTEXT,        /* Object 8  */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[4],
0x0002, 0x0805, 0x0006, 0x0001,
10, -1, -1, G_FTEXT,       /* Object 9  */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[5],
0x0002, 0x0806, 0x0006, 0x0001,
11, -1, -1, G_FTEXT,      /* Object 10  */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[6],
0x0002, 0x0807, 0x0006, 0x0001,
12, -1, -1, G_FTEXT,      /* Object 11  */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[7],
0x0002, 0x0808, 0x0006, 0x0001,
3, -1, -1, G_FTEXT,       /* Object 12  */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[8],
0x0002, 0x0809, 0x0006, 0x0001,
CHUNK, 14, 22, G_BOX,/* Object 13 SCSI */
NONE, NORMAL, (LONG)0x00FF1101L,
0x000D, 0x0806, 0x000A, 0x000B,
SCSINUM, -1, -1, G_STRING,/* Object 14  */
NONE, NORMAL, (LONG)"SCSI",
0x0003, 0x0800, 0x0004, 0x0001,
16, -1, -1, G_FTEXT,/* Object 15 SCSINUM */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[9],
0x0002, 0x0802, 0x0006, 0x0001,
17, -1, -1, G_FTEXT,      /* Object 16  */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[10],
0x0002, 0x0803, 0x0006, 0x0001,
18, -1, -1, G_FTEXT,      /* Object 17 */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[11],
0x0002, 0x0804, 0x0006, 0x0001,
19, -1, -1, G_FTEXT,      /* Object 18 */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[12],
0x0002, 0x0805, 0x0006, 0x0001,
20, -1, -1, G_FTEXT,      /* Object 19 */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[13],
0x0002, 0x0806, 0x0006, 0x0001,
21, -1, -1, G_FTEXT,      /* Object 20 */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[14],
0x0002, 0x0807, 0x0006, 0x0001,
22, -1, -1, G_FTEXT,      /* Object 21 */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[15],
0x0002, 0x0808, 0x0006, 0x0001,
SCSI, -1, -1, G_FTEXT,    /* Object 22 */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[16],
0x0002, 0x0809, 0x0006, 0x0001,
24, -1, -1, G_FTEXT,/* Object 23 CHUNK */
EDITABLE, NORMAL, (LONG)&rs_tedinfo[17],
0x0419, 0x0806, 0x0013, 0x0001,
DEFBIGSE, -1, -1, G_STRING,
                          /* Object 24 */
NONE, NORMAL, (LONG)"Sektormaximum:",
0x041C, 0x0009, 0x000E, 0x0001,
26, -1, -1, G_BUTTON,
                   /* Object 25 DEFBIGSE */
SELECTABLE|RBUTTON, NORMAL, (LONG)"512",
0x0019, 0x000B, 0x0006, 0x0001,
27, -1, -1, G_BUTTON,    /* Object 26 */
SELECTABLE|RBUTTON, NORMAL, (LONG)"1024",
0x0020, 0x000B, 0x0006, 0x0001,
28, -1, -1, G_BUTTON,    /* Object 27 */
SELECTABLE|RBUTTON, NORMAL, (LONG)"2048",
0x0027, 0x000B, 0x0006, 0x0001,
29, -1, -1, G_BUTTON,    /* Object 28 */
SELECTABLE|RBUTTON, NORMAL, (LONG)"4096",
0x041C, 0x000D, 0x0006, 0x0001,
30, -1, -1, G_BUTTON,    /* Object 29 */
SELECTABLE|RBUTTON, NORMAL, (LONG)"8192",
0x0423, 0x000D, 0x0006, 0x0001,
ABORT, -1, -1, G_BUTTON, /* Object 30  */
SELECTABLE|DEFAULT|EXIT, NORMAL,
(LONG)"OK",
0x001A, 0x0810, 0x0009, 0x0001,
0, -1, -1, G_BUTTON,/* Object 31 ABORT */
SELECTABLE|EXIT|LASTOB, NORMAL,
(LONG)"ABBRUCH",
0x0024, 0x0810, 0x0009, 0x0001,

};

OBJECT *rs_trindex[] =
{ &rs_object[0],    /* Tree 0 DIALOG */

};
```

```
/* Resource Datei Indizes für AHDICONF */

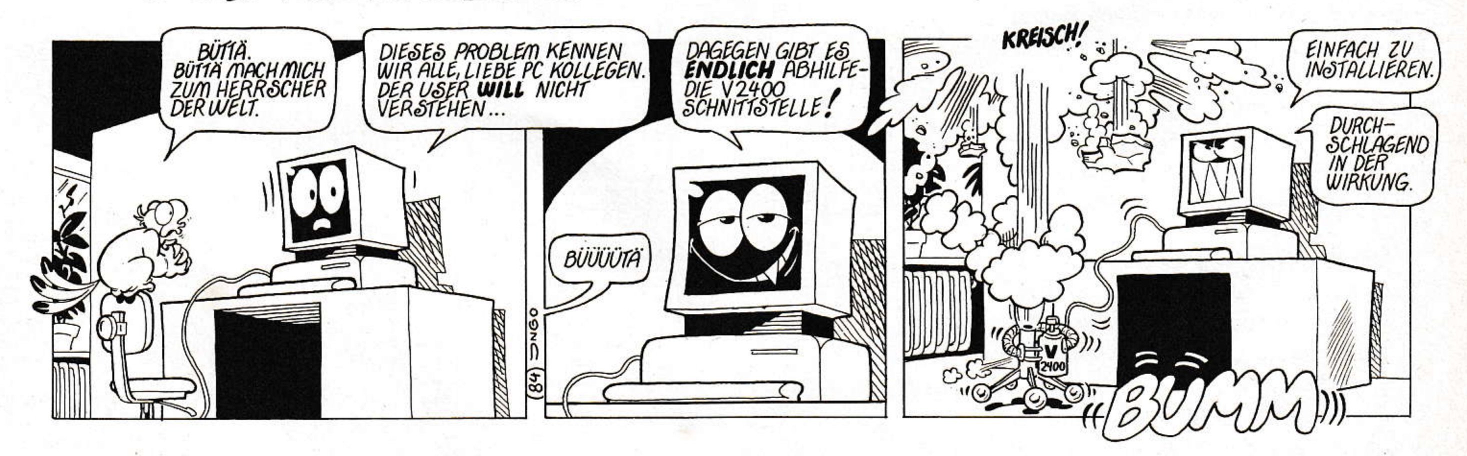
#define DIALOG    0    /* Formular/Dialog */
#define ACSINUM   5    /* FTEXT in Baum DIALOG */
#define SCSI      13   /* BOX in Baum DIALOG */
#define SCSINUM   15   /* FTEXT in Baum DIALOG */
#define CHUNK     23   /* FTEXT in Baum DIALOG */
#define DEFBIGSE 25    /* BUTTON in Baum DIALOG */
#define ABORT     31   /* BUTTON in Baum DIALOG */
```

```
/* GEM C Source Header Datei */

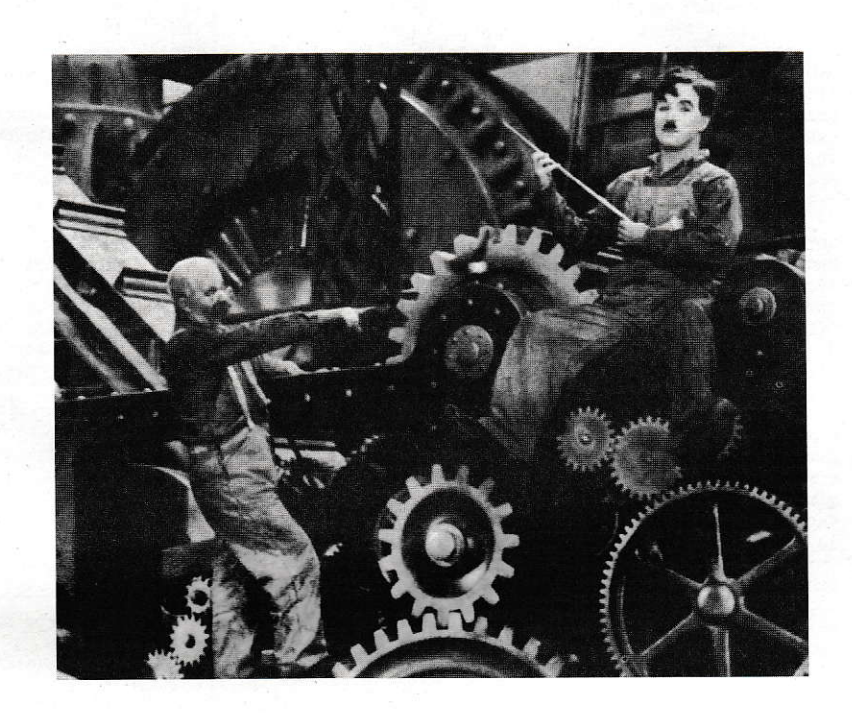
EXTERN TEDINFO rs_tedinfo[];
EXTERN ICONBLK rs_iconblk[];
EXTERN BITBLK  rs_bitblk[];
EXTERN BYTE    *rs_frstr[];
EXTERN BITBLK  *rs_frimg[];
EXTERN OBJECT  rs_object[];
EXTERN OBJECT  *rs_trindex[];

#define NUM_OBS  32
#define NUM_TREE 1
```

Listings 3 & 4: Die Header-Dateien



Der ST wird handgreiflich



Ein Port für alle Fälle!

Teil 1

Im Umfeld eines Computers gibt es immer eine Menge zu messen oder zu steuern, einfaches Umschalten zwischen verschiedenen Monitoren oder gar eine komplexe Lichtanlage. Es ließe sich nahezu alles mit dem Computer steuern, wenn der ATARI nur die nötigen Anschlüsse hätte! Einen universellen Port nämlich, ähnlich dem Userport des C-64.

Der in dieser Ausgabe vorgestellte Userport für den Atari kann sogar etwas mehr. Er ver fügt über 16 I/O- Leitungen, jede einzeln als Eingang oder Ausgang konfigurierbar, einen zusätzlichen Timer und etwas RAM. Zusätzlich werden noch ein Busmonitor/-treiber zur Anzeige der Pegel auf den I/O-Leitungen sowie das nötige Testprogramm vorgestellt. In der nächsten Ausgabe folgt als erste Anwendung des Userports die Realisierung einer parallelen Druckerschnittstelle.

Im Gegensatz zu den üblichen Bauanleitungen wird keine der vorhandenen Schnittstellen, etwa der ROM-Port, umfunktioniert, sondern der Userport zusätzlich bereitgestellt. Als Anschluß an den Rechner dient der Megaslot. Diese Art der Einbindung bietet mehrere Vorteile:

- Die Register des Port-Bausteins liegen direkt im Arbeitsspeicher der CPU (memory mapped I/O), was die Programmierung des Bausteins sehr vereinfacht.
- Durch die Interrupt-Fähigkeiten am Mega-Slot erhält der Anwender ein wichtiges Werkzeug zur Steuerung der Vorgänge am Port.
- Der Userport steht, wie die anderen Schnittstellen, an der Rückseite des Rechners zur Verfügung.
- Die Platine bietet ein zusätzliches Lochrasterfeld zum Aufbau eigener Schaltungen, deren Ausgänge ebenfalls als Schnittstelle an der Rückseite herausgeführt werden oder Steue-

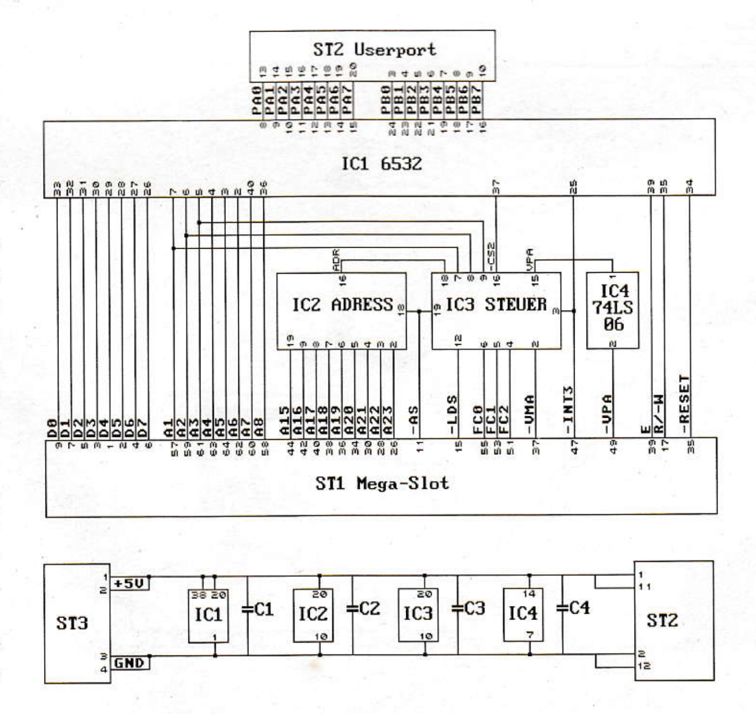


Abb. 1: Der Schaltplan für die Mega-Slot-Karte

rungsaufgaben im ST übernehmen (Cache, HD o.ä).

- Die Schnittstellen stehen weiter für ihre normale Anwendung zur Verfügung.
- Zusatzgehäuse und wackelige Steckverbindungen entfallen weitestgehend.

Als Port-Baustein findet der 6532 Verwendung. Er ist recht universell einsetzbar und läßt sich mit wenig zusätzlicher Logik in den Atari eingliedern. So war es möglich, die gesamte Steuerschaltung mit nur zwei GAL-ICs zu realisieren. Das dritte IC dient lediglich als Treiber.

Der Port-Baustein selbst wurde als Peripherie zur 6502-CPU entwickelt, stammt also aus einer IC-Familie mit 8-Bit-Datenbus. Im Gegensatz dazu ist der Datenbus im ST jedoch 16 Bit breit. Die CPU ist mit speziellen Mechanismen ausgerüstet, die neben dem normalen 16-Bit-Zugriff auch den Zugriff auf 8-Bit-Bausteine erlauben. Dabei bleibt jedoch der halbe Datenbus unbenutzt, was den Datendurchsatz verringert. Welcher Teil des Datenbusses (Low- oder Highbyte) dabei verwendet wird, meldet die CPU über -LDS und -UDS (Lower and Upper Data Strobe). Diese Signale entsprechen zusammen etwa dem A0-Signal der 8-Bit-CPU. Eine Folge dieser Aufteilung ist, daß der Baustein lediglich jedes zweite Byte im Adreßbereich belegen kann.

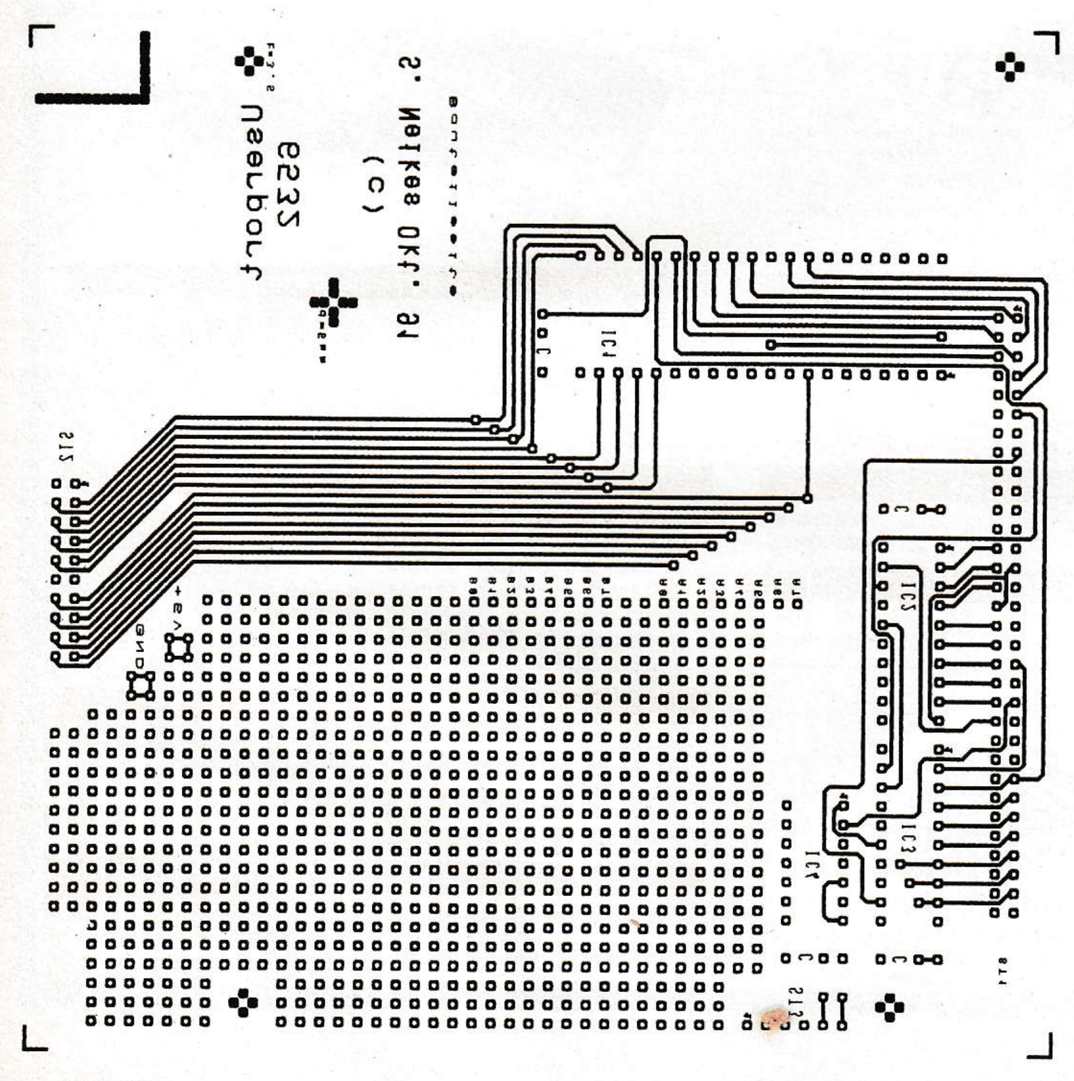
Auch bezüglich der Zugriffsgeschwindigkeit kennt die 68000-CPU zwei Betriebsarten, den synchronen und den asynchronen Speicherzugriff. Bei synchronem Betrieb drosselt die CPU ihr Tempo soweit, daß garantiert alle Bausteine am Bus mithalten können. Asynchron betrieben, muß der jeweils angesprochene Baustein melden, wann er die Daten übernommen oder bereitgestellt hat. Die Geschwindigkeit des Zugriffs wird also durch den jeweiligen Baustein bestimmt und kann so maximal werden. Die CPU kann ständig zwischen beiden Zugriffsarten wechseln. Der 6532 unterstützt leider den asynchronen Anschluß nicht und wird so, um die Schaltung einfach zu halten, synchron betrieben.

Wie im Schaltplan (Abb.1) zu erkennen, ist ein GAL, 'ADRESS', für die Adreßdecodierung zuständig. Es bestimmt, in welchem Adreßbereich die CPU den 6532 im Speicher ansprechen kann. Atari schlägt laut [1] die Bereiche $C0 0000... $CF FFFF, $FF 0000... $FF 7FFF und $FF FE00 ... $FF FFFD vor. Bei der einfachen Adreßdecodierung benötigt der 6532 $1FF Bytes Speicherraum, zuviel für den dritten Bereich. Welcher der anderen Bereiche mit dem 6532 belegt wird, ist mehr oder weniger Geschmacksache.

Das im Listing beschriebene GAL und die Programme unterstützen den zweiten Bereich. Dazu bildet es aus den Leitungen A23-A15 und -AS ein Freigabesignal. Da die Leitungen A14-A9 unberücksichtigt bleiben, erscheinen die Register des 6532 mehrfach gespiegelt im Speicher, d.h. die Register wiederholen sich mit einem Offset von $200 (s. Programmierung des 6532).

Das zweite GAL, 'STEUER', generiert die weiteren Signale für den Zugriff auf den 6532. Es schaltet die CPU auf synchronen Speicherzugriff um. Weiter beinhaltet es die Logik zur Interrupt-Bestätigung und prüft anhand des Function-Codes der CPU, ob ein erlaubter Speicherzugriff auf den Port-Baustein stattfindet, um diesen dann freizugeben.

Erlaubt sind in diesem Zusammenhang alle Datenzugriffe. Sie haben die Freigabe



Userport-Bestückungsseite

des Bausteins zur Folge. Dabei bleibt unberücksichtigt, ob im User- oder Supervisormodus auf den Speicher zugegriffen wird.

Liegt bei gültiger Adreße der Function-Code für eine Interrupt-Bestätigung vor, wird der 6532 nicht freigegeben. Stattdessen wird geprüft, ob er den Interrupt ausgelöst hat, und dies wird gegebenenfalls der CPU bestätigt.

Dazu muß zum einen bekannt sein, welchen Interrupt (Level 1-7) die CPU erkannt hat, und zum anderen, ob die -IRQ Leitung des 6532 aktiv ist.

Die CPU legt die Nummer des erkannten Interruptlevels auf die Leitungen A1-A3. Stimmt diese Nummer mit der im GAL für den 6532 festgelegten Level (Level 3) überein, und ist das -IRQ-Signal aktiv, erfolgt die Bestätigung durch das GAL über das -VPA-Signal. Es kann an diversen Stellen im Atari unabhängig voneinander erzeugt werden. Da jedoch der Zusammenschluß mehrerer TTL-Ausgänge nicht zulässig ist, muß das -VPA-Signal über einen Open-Kollector-Treiber, das dritte IC, geführt werden.

Die Platine für die Schaltung ist so ausgelegt, daß der Steckverbinder für den Userport an der Rückseite des Mega ST zur Ver-fügung steht. Er kann in die dafür vorgesehene Abdeckung eingearbeitet werden. Der Steckverbinder ist so beschaltet, daß bei Anschluß über Flachbandkabel eine Aufteilung in zwei 8-Bit-Gruppen durch einfaches Teilen des Kabels möglich ist. Die so entstandenen Anschlüsse sind in der Belegung identisch, werden aber unter anderen Adressen angesprochen.

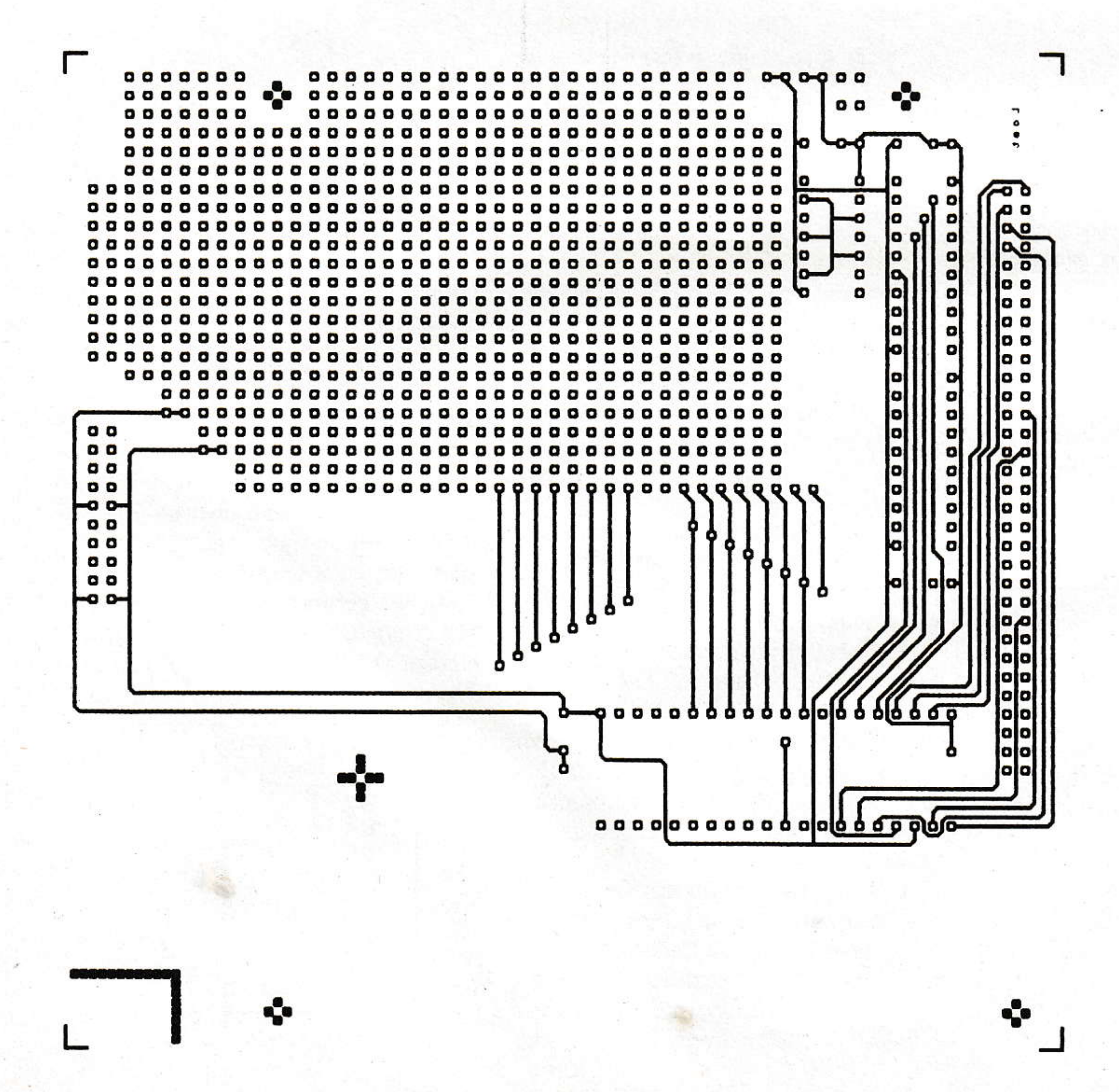
Neben den 16 I/O-Leitungen steht am Userport auch die +5V-Leitung des Rechners zur Verfügung. Sie ist wegen der Teilbarkeit ebenfalls doppelt ausgeführt und kann als Versorgungsspannung für kleinere Interface-Schaltungen genutzt werden. Beim Entwurf dieser Erweiterungen ist darauf zu achten, daß diese das Netzteil des Rechners nicht überlasten.

Das Lochrasterfeld auf der Platine kann ebenfalls für eigene Erweiterungen, etwa Treiber für die Verwirklichung einer Normschnittstelle, genutzt werden. Alle Signale des nach außen geführten Userports liegen auch am Rande des Lochrasterfeldes an.

Zur Kontrolle der Buspegel und als Testhilfe kann der im Kasten beschriebene Busmonitor dienen. Bei geeigneter Beschaltung ist er auch als Treiber für die als Ausgang arbeitenden Leitungen zu verwenden. Die maximale Leistung hängt von den verwendeten ICs und der gewählten Spannungsversorgung ab.

Der Aufbau des Userports auf der vorgefertigten Platine ist recht einfach. Bestückt wird in der üblichen Reihenfolge von passiv zu aktiv. D.h. zuerst werden der Steckverbinder ST2 und die IC-Sockel eingelötet, dann die Blockkondensatoren und das Kabel für die Stromversorgung. Als letztes wird der Megabusstecker ST1 auf der Lötseite der Platine eingesetzt. Damit der Userport den richtigen Abstand zur Platine des ST bekommt, wird eine Buchse der Bauform DIN 61412 F B ww eingesetzt. Sie muß so festgelötet werden, daß der Abstand zwischen Platine und oberem Ende der Buchse 19mm beträgt. Die Anschlüsse dieser Buchse sind länger als die der Normalausführung und erlauben diese Montage. Ist die Platine einer Sichtkontrolle auf Kurzschlüsse unterzogen, kann ein erster Test erfolgen.

Nach Öffnen des Mega-ST (Achtung: Garantieverlust!) wird die Platine provisorisch in den Slot gesteckt und der Rechner (ohne angeschlossene Peripherie, das



Userport-Lötseite

Stückliste

Userport:

IC1	6532
IC2	GAL 16V8
IC3	GAL 16V8
IC4	74LS06
C1	100nF
C2	100nF
C3	100nF
C4	100nF
ST1	64pol. DIN 41612 B female, wire wrap
ST2	Pfostensteckverbinder Stiftleiste 20 Pol 90GD
1	IC-Sockel 40pol.
2	IC-Sockel 20pol.
1	IC-Sockel 14pol.
6	15cm Litze
1	Buchsenleiste 6pol. einreihig für die Stromversorgung
Platine	
Abstandsbolzen	
Schrauben	

Busmonitor:

IC1-3	74LS04 oder 74LS06 (s.Text)
C1-C3	100 nF
16*R	220 Ω
16*D	Leuchtdiode d:3mm
BS1-BS3	Stiftleiste 20pol. zweireihig
BV1	Buchsenleiste 6pol. einreihig
3	IC-Sockel 14-Pol
S1,S2	Mikro-Schiebeschalter (1 x UM)
Platine	

Busleitung:

1	Pfostensteckverbinder Buchse 20pol. (s.Text)
1	20pol. Flachbandkabel

```
Projekt:    Userport
Gal:        ADRESS 3.0
Datum:      30.10.91
Funktion:   Decodierung des Bereichs $FF 0000 bis $FF 7FFF

%ID
   ADRESS30

%TYP
   GAL16V8

%PINS
   NC    A23    A22    A21    A20    A19    A18    A17    A16
   NC    NC     NC     NC     NC     AO     NC     -AS    A15

%LOGIC

   AO = A23 * A22 * !A15 * A21 * !-AS * A20 * A19 * A18 * A17 * A16;

%END
```

GAL-Listing für GAL-IC2

```
Projekt:    Userport
Gal:        STEUER 5.1
Datum:      30.10.91
Funktion:   Steuer und Interruptlogik

%ID
   STEUER51
%TYP
   GAL16V8

%PINS
   NC    -VMA   -IN3   FC2    FC1    FC0    A1     A2     A3
   NC    -LDS   NC     NC     --VPA  -CS2   NC     AO     -AS

%LOGIK

  !-CS2 = AO * !CF1 * FC0*!-LDS*!-VMA;

 !--VPA = AO * !-LDS
          + !-IN3 * !-AS * FC2 * FC1 * FC0 * A1 * A2 * !A3;
%END
```

GAL-Listing für GAL-IC3

ist billiger) eingeschaltet. Meldet er sich ordnungsgemäß, kann bei ausgeschaltetem Rechner auch die Stromversorgung des Userports auf den dafür vorgesehenen Stecker im Mega ST gesteckt werden. Er liegt neben dem, der die Verbindung zum Netzteil des ST herstellt.

Ein weiterer Test sollte nun ebenfalls positv verlaufen. Bootet der Rechner nicht, liegt dies wahrscheinlich an einem Kurzschluß zwischen den einzelnen Signal- oder aber zwischen Signal- und Versorgungsleitungen. Eine weitere Sichtkontrolle oder das Nachmessen aller Leitungen anhand des Schaltplans bringt den Fehler zutage.

Sind soweit alle Probleme ausgeräumt, können Sie die Platine mit den ICs bestükken und erneut testen, ob der Rechner bootet. Ist dies nicht der Fall, hat sich wahrscheinlich eine kalte Lötstelle eingeschlichen und stört den Datenfluß. Ist diese nicht offensichtlich, wird es wohl das Schnellste sein, alle Lötstellen nachzulöten, nachdem man die ICs aus den Fassungen genommen hat.

Mit dem Busmonitor und dem Testprogramm lassen sich die 16 I/O-Leitungen, das RAM und der Timer auf ihre einwandfreie Funktion prüfen. Verlief auch dieser Test zufriedenstellend, können Sie die Platine mit entsprechenden Abstandsbolzen sichern und den Rechner endgültig zuschrauben.

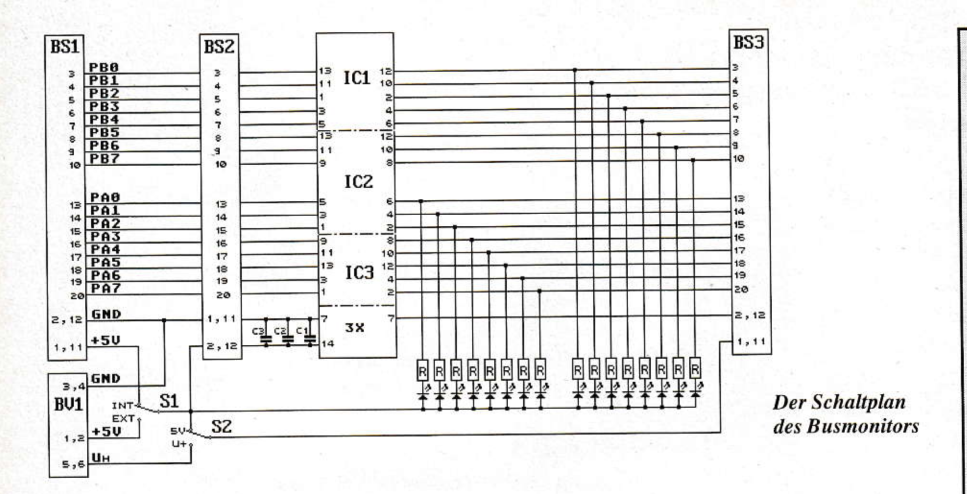
Als erstes Beispiel für die Anwendung des Userports dient das kurze BASIC-Programm. Es liest die DATA-Zeilen ein und stellt die dem Bitpattern der Zahlen entsprechenden Lichtmuster auf dem Busmonitor dar. Das Programm zeigt die Gründzüge der Programmierung des 6532 und ist leicht zu erweitern.

Der Busmonitor

Der hier vorgestellte Busmonitor hat zwei Aufgaben. Er soll zum einen als Testhilfe für den Aufbau des Userports dienen und zum anderen während des Betriebs die optische Überwachung der 16 I/O-Leitungen ermöglichen. Dazu kann er an BS1 einfach auf das Verbindungskabel zwi-

+5V	1	2	GND
B0	3	4	B1
B2	5	6	B3
B4	7	8	B5
B6	9	10	B7
+5V	11	12	GND
A0	13	14	A1
A2	15	16	A3
A4	17	18	A5
A6	19	20	A7

Die Belegung des Userports



Der Schaltplan des Busmonitors

```
REM (c)1991 by MAXON-Computer
REM ***************************
REM *      Demo-programm      *
REM * für den 16-Bit Userport *
REM *     Stephan Neikes      *
REM ***************************
.
REM ***************************
REM *   Konstanten setzen     *
REM ***************************
ram=&HFF0000
basis=&HFF0100
a=0
b=1
DIM muster(8,2)
READ musterzahl
tpause=3
.
REM ***************************
REM *     Initialisierung     *
REM ***************************
SPOKE basis+3,&HFF
SPOKE basis+7,&HFF
.
FOR muster=1 TO musterzahl
  .
  REM ***************************
  REM *      Muster lesen       *
  REM ***************************
  READ max
  READ zeit
  FOR port=0 TO 1
    FOR y=0 TO 7
      READ muster(y,port)
    NEXT y
  NEXT port
  .
  REM ***************************
  REM * Musterfolge ausgeben    *
  REM ***************************
  FOR faktor=1 TO max
    FOR x=0 TO 7
      dummy=PEEK(basis+11)
      SPOKE basis+1,muster(x,0)
      SPOKE basis+5,muster(x,1)
      SPOKE basis+47,zeit
      WHILE (PEEK(basis+11) AND 128)=0
      WEND
    NEXT x
  NEXT faktor
NEXT muster
.
REM ***************************
REM *      Rücksetzung        *
REM ***************************
SPOKE basis+3,&H0
SPOKE basis+7,&H0
.
END
```

→

schen Userport und zu steuerndem Gerät aufgesteckt werden. Zur Entlastung des Netzteils im Atari ermöglicht die Busmonitorplatine auch den Anschluß einer externen Spannungsquelle.

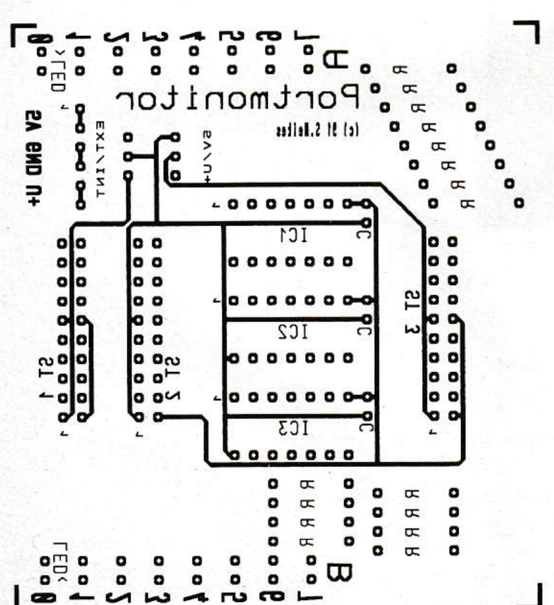
Das Schaltungsprinzip ist denkbar einfach. Die 16 I/O-Leitungen werden über invertierende Treiber an LEDs geschaltet, die wiederum über einen Vorwiderstand an +5V geführt sind. Sobald am Eingang des Treibers eine log.'1' anliegt, geht der Ausgang des Treibers auf 0V. Es fließt ein Strom durch Vorwiderstand und Diode, die Diode leuchtet.

Als Treiber können aufgrund ihrer gleichen Anschlußbelegung Bausteine der Typen 74LS04 und 74LS06 verwendet werden. Zur einfachen Pegelkontrolle bietet sich der 74LS04 an. Er liefert mit max. 25mA je Ausgang ausreichend Strom für die einfache LED-Schaltung.

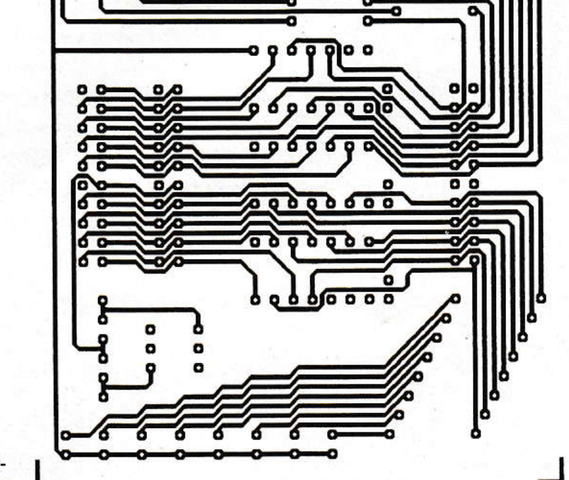
Der 74LS06 ist ein sog. Open-Kollektor-Treiber. Er ist in der Lage Spannungen bis zu 30V auf Massepegel zu ziehen. Mit ihm können kleine Lampen, Relais und sogar Schrittmotoren gesteuert werden. Nach [3] liegt dabei der maximale Strom pro o.k.-Ausgang bei 40 mA, was einer Maximalleistung von 1.2 Watt entspricht. Diese verstärkten Ausgänge stehen an BS3 zur Verfügung und dürfen auf keinen Fall extern als Eingang beschaltet werden. Die Versorgungsspannung an BS3 läßt sich über S2 wählen. In Stellung '5V' liegt die über S1 gewählte +5V-Quelle an, bei 'U+' die an BV1 Pin 1,2 anliegende Spannung.

Die Signale vom Eingang BS1 stehen auch an BS2 zur Verfügung. Er entspricht also genau dem Userport. Entsprechend der Programmierung sind die 16 Leitungen Ein- oder Ausgänge. Der einzige Unterschied zum Userport besteht darin, daß sich die Versorgungsspannung über S1 einstellen läßt. 'Int' schaltet die 5V des Atari durch, 'Ext' die an BV1 anzulegenden 5V. Da in dieser Schalterstellung auch die Treiber über BV1 versorgt werden, sollten die 5V möglichst genau eingehalten werden.

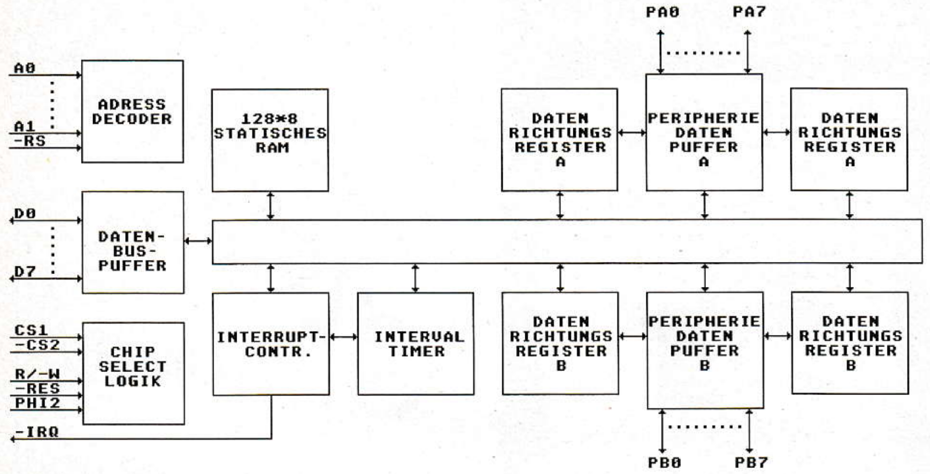
Zum Aufbau wird die Platine der Reihe nach mit IC-Sockeln, den drei Steckerleisten, zwei Schaltern sowie den Blockkondensatoren bestückt. Es



Busmonitor-Bestückungsseite



Busmonitor-Lötseite



Das Blockbild zeigt den einfachen Aufbau des 6532

folgen die Vorwiderstände und die Dioden (auf die Polung achten). Nach der Sichtkontrolle können die ICs in die Sokkel gesteckt werden.

Für die folgenden Tests müssen Sie +5V an BV1 anlegen und S1 auf 'Ext' stellen. Nun ziehen Sie die einzelnen Eingänge über einem 4.7-kΩ Widerstand auf Masse. Erlöschen die entsprechenden LEDs, ist alles in Ordnung und der Portmonitor für den Userport fertig.

Die Programmierung des 6532

Alle Register des 6532 liegen an den in der Tabelle angegebenen Adressen im Speicherbereich des Atari. Bedingt durch die 8-Bit-Struktur des Bausteins bleibt jede zweite Adresse unbelegt. Die 128 Byte RAM können durch einfaches Lesen ober Schreiben (Byte-Zugriff) angesprochen werden.

Die 16 I/O Leitungen werden, in zwei 8-Bit-Ports PA und PB aufgeteilt, über die Register ORA (Output Register), DDRA (Data Direction Register), ORB und DDRB verwaltet. Die Register DDRA/B legen die Arbeitsrichtung der einzelnen Signalleitungen fest. Wird ein Bit in diesen Registern gesetzt, arbeitet die entsprechende Leitung als Ausgang, und die in ORA/B geschriebenen Daten werden ausgegeben. Liest man die Register ORA/B, so gibt der 6532 die dem logischen Pegel des Ports entsprechenden Daten an den Rechner weiter.

Die Ports PA und PB unterscheiden sich lediglich durch ihre Ausgangstreiber. PA hat TTL-kompatible Treiber, während PB sogenannte 'push-pull'-Treiber hat. Mit ihnen kann er bis zu 3mA an 1,5V liefern, genug, um direkt Transistoren anzusteuern.

Die Betriebsart des Timers wird durch die Adresse des Timer-Registers festgelegt, in das der Timer-Startwert (1-255) eingeschrieben wird. Der bei den Registern angegebene Faktor legt fest, nach wievielen Takten der Leitung E (800Hz) der Registerinhalt erniedrigt wird. Ob der Timer beim Erreichen der Null einen Interrupt auslößt, wird ebenfall durch Wahl des Timer-Registers bestimmt. Dieser Nulldurchlauf des Timers setzt immer das 7.Bit im Interrupt-Flag-Register. Ein gesperrter Interrupt verhindert lediglich die Aktivierung der -IRG-Leitung und somit die Unterbrechung der CPU.

Auch eine Flankenerkennung an PA7 wird in jedem Fall im Interrupt-Flag-Register (Bit 6) vermerkt. Ob dadurch ein Interrupt ausgelößt wird, legt der Programmierer durch Zugriff auf das jeweilige 'EDGE'-Register fest.

Bei der Initialisierung des Bausteins ist zu beachten, daß durch einen Reset zwar die Interrupts gesperrt werden, doch während des Resets durch ungünstige Pegel an PA7 das 6.Bit im Interrupt-Flag-Register gesetzt werden kann. Zur Vermeidung von Fehlern im Programmablauf sollte es daher durch Auslesen gelöscht werden.

Weitere Informationen zu diesem Baustein können Sie den entsprechenden Unterlagen der Hersteller entnehmen.

Stephan Neikes

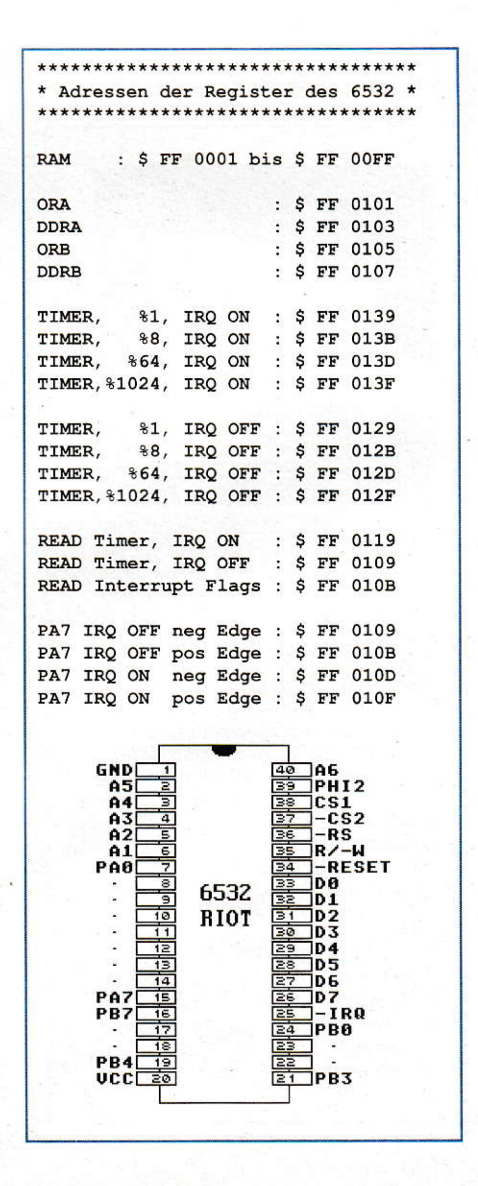
Literatur:

[1] Jankowski, Reschke, Rabich; Atari ST Profibuch; 1989 Düsseldorf, Sybex-Verlag

[2] Motorola; M68000 User Manual; 1989 Englewood Cliffs N.J., Prentice Hall

[3] Texas Instruments, Pocket-Guide, Band 1: Digitale integrierte Schaltungen


```
60: .
61: REM Maximale Musterzahl
62: .
63: DATA 7
64: .
65: REM Anzahl der Wiederholungen;
    Wartezeit; Muster je 8*A / 8*B
66: .
67: DATA 1,3
68: DATA 01,02,04,08,16,32,64,128
69: DATA 128,64,32,16,8,4,2,1
70: .
71: DATA 1,7
72: DATA 01,03,07,15,31,63,127,255
73: DATA 01,03,07,15,31,63,127,255
74: .
75: DATA 1,15
76: DATA 255,127,63,31,15,7,3,1
77: DATA 255,127,63,31,15,7,3,1
78: .
79: DATA 1,31
80: DATA 01,03,07,15,31,63,127,255
81: DATA 01,03,07,15,31,63,127,255
82: .
83: DATA 4,63
84: DATA 255,127,63,159,207,231,243,121
85: DATA 255,127,63,159,207,231,243,121
86: .
87: DATA 1,127
88: DATA 60,30,15,7,3,1,0,128
89: DATA 60,30,15,7,3,1,0,128
90: .
91: DATA 3,255
92: DATA 192,96,48,24,12,6,3,1
93: DATA 192,96,48,24,12,6,3,1
```



```
**********************************
* Adressen der Register des 6532 *
**********************************

RAM    : $ FF 0001 bis $ FF 00FF

ORA                    : $ FF 0101
DDRA                   : $ FF 0103
ORB                    : $ FF 0105
DDRB                   : $ FF 0107

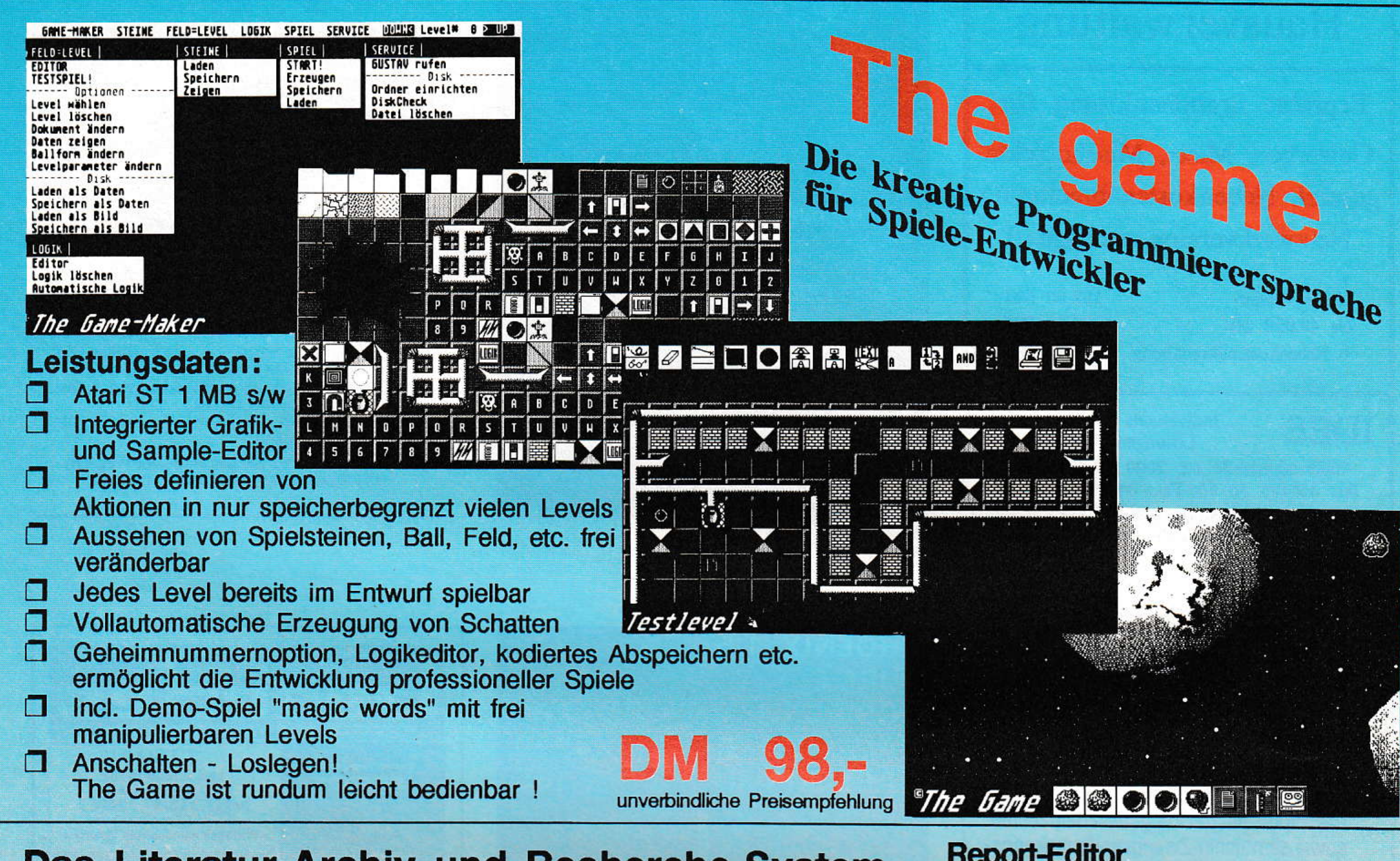
TIMER,    %1, IRQ ON   : $ FF 0139
TIMER,    %8, IRQ ON   : $ FF 013B
TIMER,   %64, IRQ ON   : $ FF 013D
TIMER, %1024, IRQ ON   : $ FF 013F

TIMER,    %1, IRQ OFF  : $ FF 0129
TIMER,    %8, IRQ OFF  : $ FF 012B
TIMER,   %64, IRQ OFF  : $ FF 012D
TIMER, %1024, IRQ OFF  : $ FF 012F

READ Timer, IRQ ON     : $ FF 0119
READ Timer, IRQ OFF    : $ FF 0109
READ Interrupt Flags   : $ FF 010B

PA7 IRQ OFF neg Edge   : $ FF 0109
PA7 IRQ OFF pos Edge   : $ FF 010B
PA7 IRQ ON  neg Edge   : $ FF 010D
PA7 IRQ ON  pos Edge   : $ FF 010F
```

Software-Neuheiten



Leistungsdaten:

- ❒ Atari ST 1 MB s/w
- ❒ Integrierter Grafik- und Sample-Editor
- ❒ Freies definieren von Aktionen in nur speicherbegrenzt vielen Levels
- ❒ Aussehen von Spielsteinen, Ball, Feld, etc. frei veränderbar
- ❒ Jedes Level bereits im Entwurf spielbar
- ❒ Vollautomatische Erzeugung von Schatten
- ❒ Geheimnummernoption, Logikeditor, kodiertes Abspeichern etc. ermöglicht die Entwicklung professioneller Spiele
- ❒ Incl. Demo-Spiel "magic words" mit frei manipulierbaren Levels
- ❒ Anschalten - Loslegen! The Game ist rundum leicht bedienbar !

DM 98,-
unverbindliche Preisempfehlung

Das Literatur-Archiv und Recherche-System Für alle Atari ST und TT Computer

Review

Das Bedienungskonzept
- ❒ Menüsteuerung; Pull-Down-Menüs; Tastatursteuerung
- ❒ Relationales Datenbanksystem optimale Durchgängigkeit
- ❒ Hohe Effektivität Geringe Einarbeitungszeit, Schnelligkeit, Zuverlässigkeit

Dateneingabe-Kontrollen
- ❒ Dateneingabe in vorgefertigte Masken
- ❒ Duplikatprüfung für Autoren, Listen und Datensätze
- ❒ Vollständigkeitsüberprüfung

Suche
- ❒ Volltextrecherche (global/lokal)
- ❒ Manuelle Datensatz-Auswahl
- ❒ Katalogsuche

Import/Export-Editor
- ❒ Datenexport in beliebig gestaltbaren Formaten
- ❒ Datenimport-Möglichkeiten
- ❒ Datensicherung in ASCII
- ❒ Laden/Speichern von Export-Formaten
- ❒ Duplikatsprüfung beim Import (autom.)

Einsatzbereiche
- ❒ Entwicklungs-u. Forschungsabteilungen
- ❒ Universitäten, Tagungsbüros, Lehrer
- ❒ Juristen, Ärzte, Studenten, Projektleiter

Report-Editor
- ❒ Reporte in beliebig gestaltbaren Formaten
- ❒ Laden/Speichern von Report-Formaten
- ❒ Schneller Listendruck aller Kataloge

Sonstiges
- ❒ Kompakte und flexible Datenpräsentation in Bildschirmlisten
- ❒ Frei gestaltbarer Themenkatalog
- ❒ Freie Formulierung von Druckeranpassung
- ❒ Tastaturabkürzungen wichtiger Funktionen in Dialogen und Menüs
- ❒ Incl. Konvertierungsmodul

Mehrfachbenutzung
- ❒ Installierung auf einemZentralrechner
- ❒ Datenbestand auf den Hostrechnern
- ❒ Datentransfer Export/Import via Netzwerk

System-Anforderungen
- ❒ Atari Computer der ST/TT-Serien
- ❒ S/W-Monitor (Großbildschirm optional)
- ❒ Lauffähig auf allen TOS-Versionen
- ❒ GDOS-Kompatibilität

DM 148,-
unverbindliche Preisempfehlung

Heim Verlag
Heidelberger-Landstr. 194
6100 Darmstadt-Eberstadt
Telefon 0 61 51 / 5 60 57
Telefax 0 61 51 / 5 60 59

Bitte senden Sie mir
___ The game a 98,– DM
___ ST/TT-Review a 148,– DM
Name, Vorname ________
Straße ________
PLZ,Ort ________

zuzüglich 6,--DM Versandkosten (Ausland 10,–DM)

unabhängig von der bestellten Stückzahl

Preise sind unverbindlich empfohlene Verkaufspreise

In Österreich
Reinhart Temmel
GmbH & Co.KG
St.Julienstr. 4a
A-5020 Salzburg

In der Schweiz
DTZ Data Trade AG
Landstraße 1
CH-5415 Rieden-Baden



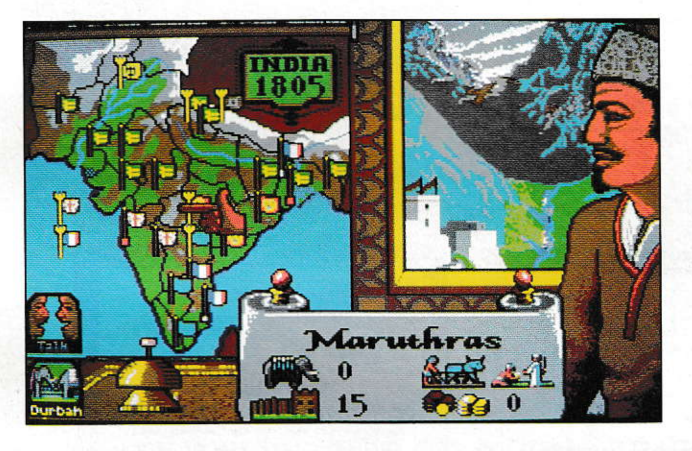
Hallo Spiele-Freaks!
Es ist auch in diesem Jahr wieder ganz deutlich zu erkennen, daß es auf die Weihnachtszeit zugeht. Nahezu alle bekannten Spielehersteller werfen eine Unmenge an Neuheiten auf den Markt. Rushware präsentiert seinen neuen Katalog mit insgesamt 17 neuen Titeln, darunter hochkarätige Vertreter wie „R-TYPE-II" oder der brandneue Flugsimulator „Flight of the Intruder". United Software hat eine neue Golfsimulation herausgebracht, die von dem bekannten Software-Haus Microprose entwickelt wurde (bekannt durch den legendären Klassiker „Microprose Soccer"). Höchste Naturtreue im Spielablauf und digitalisierte Grafiken zeichnen dieses Sportspiel aus. Auch Electronic-Arts wartet mit diversen Neuheiten auf. Es regt sich also wieder was in der Spiele-Szene. Das Sommerloch scheint überwunden!

CM

CHAMPION OF THE RAJ



Dieses Spiel des Software-Labels PSS (gebildet aus Mirrorsoft und den Programmierern von Level Nine) stellt eine etwas andere Art von Strategiespiel dar. Es ist eine Mischung zwischen einem Arcade-Game mit Action und einem typischen Strategie-Adventure, bei dem die richtigen Entscheidungen zu treffen sind. Gesteuert wird es komplett mit der Maus oder einem Joystick. Zu Beginn hat man die Möglichkeit, einen aus sechs verschiedenen Charakteren zu wählen, mit dem man sich dann ins Geschehen stürzt. Die Geschichte spielt zu Beginn des 19ten Jahrhunderts im damaligen Indien. Der Spieler muß sich z.B. als französischer Konsul betätigen und als solcher Entscheidungen über Steuern, Ausrüstung für die Armee treffen oder eventuell die Macht in allen Regionen an sich reißen. Dazu bekommt man eine Übersichtskarte, auf der alle indischen Staaten mit den entsprechenden Flaggen markiert sind. Sobald der Spielen einen



dieser Staaten wählt, wird er von dem momentanen Staatsoberhaupt eingeladen. Hier spielen sich die Arcade-Sequenzen in Form einer Tigerjagd oder eines Elefantenrennens ab. Macht man seine Sache gut, gewinnt man an Ansehen, andernfalls muß man es auskämpfen. Ebenso können Naturkatastrophen auftreten, und der Spieler hat zu entscheiden, ob er dem jeweiligen Staat finanzielle Hilfestellung leisten soll. Insgesamt ist der Aufbau eines Imperiums ein sehr langsam vonstatten gehender Prozeß, und die richtigen Entscheidungen zu treffen, ist nicht immer einfach. Man sollte stets den diversen Paraden beiwohnen, besonders, wenn man viele Elefanten besitzt. Dies wird mit einem deutlichen Anstieg des Status' belohnt.

Für die einzelnen Aktionen erscheinen entsprechende Icons. Allerdings sind nicht immer alle Aktionen erreichbar. Truppen bzw. Elefanten einkaufen oder Steuern erhöhen sind Maßnahmen, die nicht jederzeit ausgeführt werden können. Man sollte sie also nutzen.

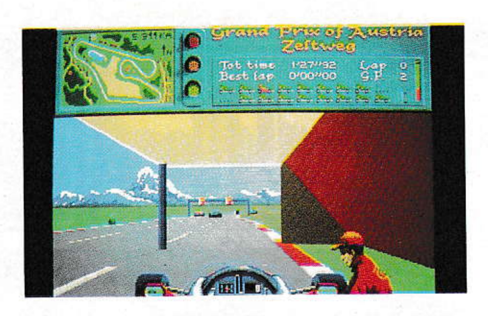
Insgesamt ist dieses farbenfrohe Strategie-/Action-Spiel eine gute Möglichkeit für den Anfänger, sich mit Spielen dieses Genres vertraut zu machen. Durch die fünf integrierten Action-Szenen kommen auch Joystick-Akrobaten auf ihre Kosten. Die einzigen Schwachpunkte sind die ermüdend langen Ladezeiten von Diskette und die etwas trägen Animationen bei den Arcade-Sequenzen. Es ist eben ein etwas anderes Spiel.

ddf/CM

VROOM



In den Rängen tobt das Publikum und jubelt den besten Fahrern zu. Die völlig verstaubten Gesichter unter den Helmen sind jetzt auch noch schweißnaß. Im Fuß auf dem Gaspedal schmerzt ein Krampf. Ein Unglücksrabe wird aus der Kurve getragen und saltiert am Pistenrand. Für Computerrennfahrer ist das alles kein Problem, sondern purer Spaß. Software-Hersteller Langhor aus Frankreich steuert jetzt mit einer Rennsimulation rasant den deutschen Markt an. Schon der Titel verheißt Tempo, qualmende Reifen, röhrende Motoren: „Vrrrroommm!" Die Formel-1-Schlitten rasen vorbei, aber die Atari-Piloten führen das Feld an. Vielleicht. Passieren kann nichts, doch schwierig bleibt das Zügeln wildgewordener Pferdestärken allemal. Mit „Vroom" wird der Atari sechs Rennen lang zum Formel-1-Cockpit. Joystick oder Maus verwandeln sich in eine Kombination aus Lenkrad, Schaltknüppel, Gaspedal, Kupplung und Bremse. Anfänger oder eher bequemliche Fahrer beschaffen sich im Hauptmenü ein Automatikgetriebe. Außerdem kann man zwischen zwei Modi wählen, dem einfacheren Arcade- und dem simulationsmäßigen Grand-Prix-Modus. Arcade, das heißt hier, im ersten Durchgang acht andere Renner zu überholen und möglichst unbeschadet viele Kilometer zu machen. Dann darf man im zweiten Lauf starten, wo man wiederum drei weitere Konkurrenten Staub schlucken läßt. Das simulierte Grand Prix orientiert sich an den tatsächlichen Regeln. In der ersten Runde muß sich jeder Fahrer qualifizieren, um mit 25 anderen Pedaltretern am Grand Prix teilnehmen zu dürfen. Nach jedem Durchgang zeigt eine Statistik an, wieviel Punkte hinzugekommen sind. Die sechs besten Rennpiloten kämpfen dann um den Weltmeisterschaftstitel. Ein Demo zeigt, wie Könner die Kurven nehmen, Gefahrensituationen bewältigen und an Konkurrenten vorbeiziehen. Geübte machen gleich ernst, Einsteiger wählen den Trainings-Modus an. In beiden Fällen sieht der Bildschirm gleich aus: im großen Bildschirmfenster eilt schnell und glatt die dreidimensional dargestellte Landschaft vorbei, oberhalb sind ein kleiner Parcoursplan und eine Anzeige für High-Scores, gedrehte Runden und Benzinmenge zu sehen. Am unteren Rand zeigt ein Tacho die Geschwindigkeit und den aktuellen Gang an. Auf dem Pistenplan bewegt sich ein weißes Pünktchen dort, wo man selber entlangrauscht; schwarze Pünktchen stehen für die mitstreitenden Wagen. Dank der schnellen Grafik meint man, gleich ins hurtig näher kommende Gebüsch zu rasen oder ein Hinweisschild abzurasieren. Das kommt vor. Zur Strafe muß man einen gewissen Streckenabschnitt noch einmal fahren und verliert wertvolle Zeit. Kleinere Blessuren am Fahrzeug führen nicht zum Spielende, bleiben aber dennoch nicht folgenlos. Das Fahrwerk verzieht sich, und ganz realistisch wird die Steuerung weniger exakt. Nach einem absolvierten Rennen läßt man am besten die Mechaniker an seine PS-Schleuder: eventuell Reifen wechseln, kaputte Blechteile austauschen und Tank auffüllen. Und dann wieder ins enge Cockpit gequetscht und losgejagt. Steht ein weiterer Atari in Reichweite, wird ein Null-Modem in den seriellen Port gesteckt, so daß zwei Freunde hoher Drehzahlen gegeneinander und gegen die computergesteuerten Flitzer antreten können. Aha, der Freund wohnt in Australien. Dann geht es auch mit dem Modem - ein wahrhaft internationales Rennereignis, das allerdings etwas teurer kommt. Ob im fernen Land der Känguruhs oder bei uns, der Sound ist der gleiche: mehr oder minder überdreht jault einem der Motor in die Ohren. Auf der Rennstrecke mag der Sound ja auch nicht angenehmer sein, aber am heimischen Schreibtisch dürfte es gern besser klingen. Schließlich streut man sich ja auch keine fein geraspelten Autoreifen um die Tastatur und parfümiert den Monitor mit Rennbenzin. Die Grafik ist weit besser als der Sound. Fast ohne zu ruckeln, kommen Randbewuchs und konkurrierende Wagen näher und entfernen sich genauso glatt. Meckern kann man trotzdem. Daß dekorative Kleinigkeiten auch eine Rennsimulation aufwerten, ist ja nicht erst seit „Lotus Esprit Challenge" bekannt. „Vroom" bietet an ansehnlichem Drumherum nicht viel. Sicher, die Steuerung ist ordentlich, über einen Mangel an Features kann man sich auch nicht beschweren. Aber die Zeit bleibt nicht stehen. Grafik wird immer wichtiger. Nicht umsonst bersten die Disketten vieler neuerer Unterhaltungsprogramme nur so von den Datenmassen prachtvoller Grafiken mit witzigen oder realistischen Einzelheiten. „Vroom" ist nicht übel, „Lotus" von Gremlin aber sowohl grafisch als auch spielerisch ausgefeilter.



CBO

TERMINATOR II



Wer den Film nicht gesehen hat, bekommt mit diesem Spiel die Möglichkeit, wenigstens einige kurze Szenen in digitalisierter Grafik zu sehen. In Terminator II wird, wie so oft, versucht, Filmszenen als Computer-Spiel umzusetzen. Dabei sind insgesamt 8 Levels zu bestehen. Haupkritikpunkt hierbei ist die Tatsache, daß die verschiednenen Levels entweder zu schwer oder zu leicht sind. Daß die Spielfigur auf die Aktionen mit dem Joystick nur träge reagiert, ist dabei auch nicht gerade hilfreich. Level eins stellt ein Duell des Terminators mit einem gleichstarken, vom Computer gesteuerten Roboter dar. Die einzige Chance, dieses Level zu überstehen, besteht darin, sich so schnell wie möglich zu bewegen und recht häufig Schläge, Tritte und Kopfstöße auszuteilen. Wie weit beide Roboter schon dem Ende nah sind, sprich: der endgültigen Zerstörung, sieht man an zwei Gesichtern, die mit zunehmender Schwäche immer grauer werden. Im nächsten Level sitzt der Terminator auf einem Motorrad und muß durch geschicktes Manövrieren möglichst viele herumliegende Bonuspunkte einsammeln. Der Spieler sieht das Szenario aus der Vogelperspektive. Die Straße ist mit allerlei Unrat bestückt, was die Sache extrem erschwert. Zudem muß der Spieler darauf achten, daß der folgende Lastwagen ihn nicht einholt, ansonsten findet dieses Level ein vorzeitiges Ende. Das dritte Level besteht aus einem Puzzle. Ein etwas durcheinandergeratenes Bild muß innerhalb einer bestimmten Zeit wieder geordnet werden. Dieses Schiebe-Puzzle ist so ziemlich die einzige faire Herausforderung in dem Spiel, später wird wieder kräftig geschossen, um in einer Helikopter-Szene bestehen zu können.



Insgesamt kann Terminator II nicht den Erwartungen an es gerecht werden. Zwar ist es grafisch recht gut präsentiert, aber die holperige Steuerung und müden Reaktionen sowie die teilweise unlösbaren Situationen lassen nicht gerade dauerhafte Spielfreude aufkommen.

ddf/CM

THE FAMOUS FIVE ON A TREASURE ISLAND



Wer erinnert sich nicht gern an die Detektivabenteuer der 'Fünf Freunde', die, egal ob als Buch oder als Fernsehserie immer für ein paar unterhaltsame Stunden gut waren? Enigma Variations hat dem Quintett sein lange überfälliges Computerspiel in Form eines Text-Adventures spendiert, bei dem vor allem die jüngeren Enid Blyton-Fans voll auf ihre Kosten kommen. Julian, Dick, Anne, Georgina, genannt George, und Hund Timmy verbringen ihre Schulferien bei Tante Fanny an der britischen Küste. Gleich in der Nähe befindet sich eine mysteriöse Insel. Ausgerüstet mit Fakkel, Streichhölzern, einem Tau sowie einer Gruppenration Sandwiches rudern die Teenager Richtung Gefahr. Auf der Insel warten ein Schiffswrack und die Ruine einer bereits vor Jahrhunderten niedergebrannten Burg auf ihre Erforschung. Es dauert nicht lange, bis die Kids einem Schmugglerring auf die Schliche kommen, der sich nebenbei auch noch mit Kidnapping beschäftigt. Als Timmy in eine Erdspalte fällt und Julian bei einem riskanten Tauchmanöver in stürmischer See beinahe absäuft, spitzt sich die Situation dramatisch zu. Famous Five on a Treasure Island hält sich genauestens an die Jugendbuchvorlage, fügt der spannenden Handlung aber noch ein paar einfache Puzzles hinzu. Die übersichtliche Aufmachung sorgt für kinderleichte Bedienung: In der oberen Bilschirmhälfte illustrieren über 30 Grafiken mit leichtem Hang zum Kitsch das Geschehen. Darunter erscheinen leicht verständliche Texte in einfachem Englisch. Wichtige Funktionen wie Spielstände sichern, Orte untersuchen oder die Wahl des gerade aktiven Charakters werden über Pulldown-Menüs aktiviert. Womit wir auch schon beim innovativsten Feature des ansonsten eher altmodischen Adventures wären: Der Spieler kann zwischen vier Akteuren umschalten, um alle Probleme bewältigen zu können. Nur Georgina kennt den Weg zur Insel, und nur der kräftige Julian ist in der Lage, den vierbeinigen Begleiter mit Hilfe des Seils aus der Erdspalte zu befreien. Schade, daß sich der Parser größtenteils auf schlichte Zwei-Wort-Kommandos beschränkt und bescheidene 50 Verben plus Synonyme versteht. Adventure-Profis haben das nicht allzu umfangreiche Rätsel locker an einem Nachmittag gelöst. Einsteiger, die vor Infocoms oder Magnetic Scrolls Textwüsten zurückschrecken, sollten dagegen unbedingt mal einen Blick riskieren.

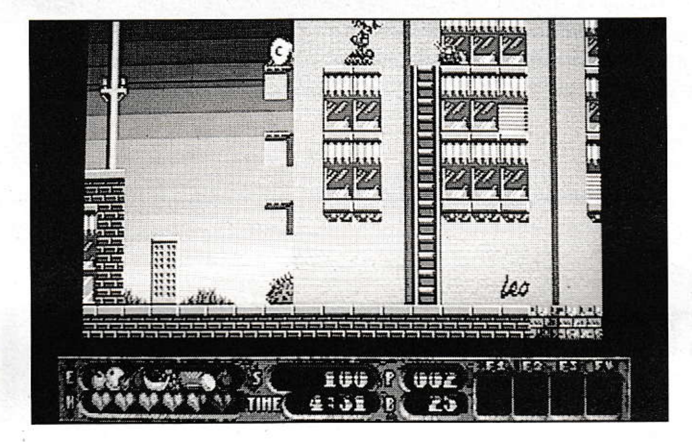


CBO

ROLLING RONNY



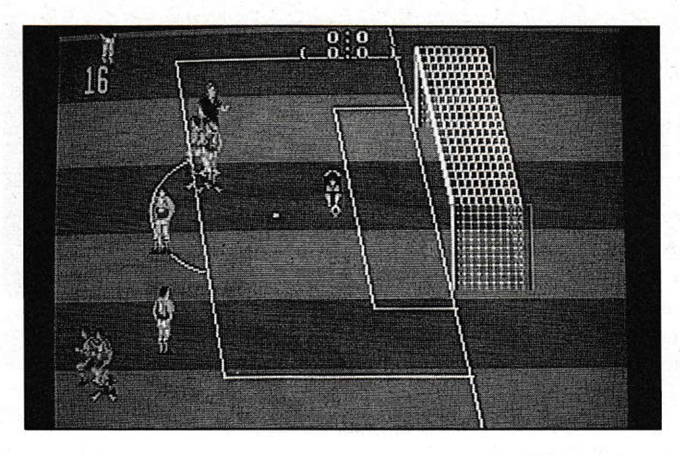
Ein Laufbursche auf Rollschuhen? Was Besseres kann sich Scotland Yard gar nicht wünschen. Da „Rolling Ronny" - so nennt man ihn - auch tapfer und wehrhaft ist, bekommt er immer die gefährlichsten Aufträge. Der Spieler von Starbytes neuem Geschicklichkeits-Game steuert den Botenjungen durch belebte Straßen. In der Tasche trägt Ronny eine geheime Nachricht, die schnellstens ihren Empfänger erreichen soll. Mit wehendem Haar rollt er die unmöglichsten Wege entlang. Der Bürgersteig wird zur Rennstrecke. Baugerüste und Treppen kommen als akrobatische Herausforderung gerade recht. Schienen überwindet Ronny genauso wie Plattformen. Währenddessen taucht übles Gesindel auf. Rocker auf Motorrädern kreuzen seinen Weg. Autofahrer, die die Stadt als Rennstrecke betrachten, kommen ihm in die Quere. Schlafmützige Fußgänger behindern die eilige Rollschuhfahrt des jungen Kuriers. Gegen sie und andere lästige Gestalten wehrt Ronny sich mit wohlgezielten Schüssen. Denn sobald er einen der Störenfriede streifen oder berühren würde, gingen ein paar Health-Punkte flöten. Genauso wichtig wie Geld und Gesundheit sind die kleinen Würfel, die in allen fünf Levels umherliegen. Die Würfel sind sozusagen die Eintrittskarte ins folgende Level. Wenn nicht alle Würfel des jeweils vorhergehenden Spielabschnitts aufgesammelt sind, sitzt man fest. Ebenso, wenn „Ronny" nicht genügend Geld für den Bus übrig hat. Also muß der rollerskatende Bubi auch Taler sammeln. Da trifft es sich gut, daß er bei seinen übermütigen Luftsprüngen manchmal unsichtbare Bonusfelder berührt, aus denen es Kleingeld regnet. Außerdem lösen sich manche der abgeschossenen Gegner umgehend in Geld auf. Des lieben Geldes wegen erledigt Ronny zwischendurch auch mal einen Aushilfsjob. Für einige Pennys ist er gern bereit, ein Päckchen zu einem Haus ein paar Straßen weiter zu bringen. Ausnahmsweise kann man in diesem Spiel auch Energie und Gesundheitspunkte für Geld bekommen - natürlich nur im Spezialgeschäft. Dort liegen auch solch irre Dinge wie Weitsprungfedern zum Verkauf. Bei solch schnellen Fortbewegungsmitteln kracht es manchmal auch. Ronny sieht dann nur noch Sterne. Blutrünstig oder brutal gerät das Spiel jedoch an keiner Stelle. Die Abenteuer des rollschuhfahrenden Ronny ereignen sich in einer bunten, lustigen Comic-Welt. Feine Musik und witzige Animationen machen Ronnys Fahrt ganz schön amüsant. Wären die Levels noch ein bißchen abwechslungsreicher, hätte aus „Rolling Ronny" ein echter Hitparadenstürmer werden können.



CBO

MANCHESTER UNITED EUROPE





Bolzen nach Maß: jetzt spielen die „großen Jungs"! Action und knallharten Fußball bietet Krisalis neue Tretersimulation „Manchester United Europe". Kein müdes Tipp-Kick, sondern ein realistisches Spiel mit Fouls, roten und gelben Karten sowie einem Schiedsrichter. Eine ausgefuchste Steuerung macht alle Finessen möglich: Volleys, hohe und flache Pässe (wie einst Netzer, „aus der Tiefe des Raumes"), Einwürfe, Freistöße und Eckbälle, alles was das Fanherz begehrt. Aber vor lauter hartem sportlichem Einsatz hat Krisalis die Denkarbeit nicht vergessen. Was ist das pulstreibendste Workout auf dem Rasen ohne die ausgefuchste taktische Planung, das strategische Duell der Meistertrainer? Der Schlüssel zum Ruhm liegt in der raffinierten Aufstellung der Mannschaften. Damit der Screen-Beckenbauer nicht den Überblick verliert, schaut er im Statistikfenster nach, wie stark der Gegner und wie gut die eigene Mannschaft ist. Das ist die Phase, in der unser Bundes-Franz immer „Schau' mer mal..." sagt: die spielerischen Eigenschaften des Gegners zu kennen, ist schon der halbe Sieg. Die nächste Vorbereitungsphase wünscht sich sicher jeder Nationalcoach: die Einstellung der Spielzeit, des Schwierigkeitsgrads und die Anzahl der Mitspieler. Ganz einfach ist das nicht, gilt es doch, die Kontrolle über die aktiven Balltreter und die Torhüter zu behalten. Wie bei den meisten Joystick-Matchen üblich, darf nur der Kicker ran, der dem Ball am nächsten ist. Noch entspricht das simulierte Gewühl auf dem Bitmap-Rasen nicht ganz der Realität; stellen Sie sich nur einmal vor, man müßte 22 Spieler gleichzeitig dirigieren! Der Joystick würde nach verbranntem Gummi riechen. Aber wofür hat man Freunde und einen Computer? Mit einem handelsüblichen Multiplayer-Adapter kann man die Steuerung der Bitmap-Bolzbande auf vier menschliche Könner verteilen, und sich - wie im richtigen Leben - über die Fehler der anderen ärgern. Die Kommentare bei einem verpatzten Einwurf sind, wie jeder Aktive weiß, das Salz in der Fußballer-Suppe. Das müssen auch die Krisalis-Programmierer gewußt haben, denn Einwürfe, Freistöße und Eckbälle werden besonders effektvoll simuliert. Der Joystick-Libero steuert ein Fadenkreuz über den Rasen und bestimmt so die Richtung des Einwurfs. Die Landung des Balles ruft gewöhnlich ein größeres Getümmel hervor (im Fachjargon „konzentriertes Spiel im Mittelfeld" genannt), bei dem der eine oder andere Körperkontakt auch beim besten Willen nicht mehr zu vermeiden ist. Hier setzt bei den programmierten Kickern die nicht zu Unrecht so genannte „Kollisionsabfrage" ein, die Krisalis allerdings etwas zu flächendeckend geraten ist: selbst wenn der Spieler um mehrere Körperlängen am anderen vorbeibrettert, ist die gelbe Gefahr im Anzug. Dem eifrig mitlaufenden Schiri sitzen die roten und gelben Karten sehr locker, und das Spiel leidet durch die eigentlich unnötigen Unterbrechungen. Aber vielleicht ist dieses vermeintliche Manko doch der Tribut an die Realität: auf diese Weise kann man sich doch wenigstens über den Mann in Schwarz prima aufregen. Auch wenn der Gegner die branchenüblichen Härtegrade überbetont und ein eigener Spieler ausscheidet, ist noch lange nicht Schluß: frisch eingewechselte Ersatzspieler sind bestens geeignet, eine solide Revanche vorzunehmen und die Chancen wieder auszugleichen. Völlig klar, daß bei einem derart actiongeladenen Geschehen die Kulisse im Stadion entsprechend ist. Die Fans machen sich lautstark bemerkbar, stören aber die Konzentration der Firebutton-Torjäger nicht nachhaltig. Wie in den besten Worldcup-Spielen wird die Atmosphäre durch Musik sehr dicht und realistisch. Der Sound wirkt auch nach mehreren Stunden heißen Bolzens nicht unangenehm und störend. Die Programmierung von „Manchester United Europe" darf als gelungen bezeichnet werden: die Menüs sind übersichtlich und auch in der Hektik des Geschehens schnell und sicher zu bedienen. Der einzige Schwachpunkt ist die bereits erwähnte „übersensible" Kollisionsabfrage bei Fouls. Die dadurch geforderte „körperlose" Spielweise paßt eigentlich nicht zu dem sonst sehr actionbetonten Spiel. Die Steuerung ist sicher nicht einfach und für die Dramatik des Kampfes vielleicht zu komplex. Andererseits ist diese Gewöhnungsbedürftigkeit eher eine zusätzliche Herausforderung für ambitionierte Computerbolzer. Auch wenn das Game nicht ganz an die Klasse von „Emlyn Hugh's International Soccer" und „Kick off" herankommt, so gehört „Manchester United Europe" trotzdem zu den Spitzenprogrammen unter den Fußballspielen.

CBO

MAGIC POCKETS



Man kommt nicht darum herum, zuzugeben, daß die cleveren Jungs der Bitmap Brothers ein weiteres Spiel veröffentlicht haben, das sich anschickt, die Game-Charts zu stürmen. Magic Pockets sieht durch die metallic-blaue Tönung ein wenig dem Spiel GODS ähnlich. Das Hintergrund-Scrolling sowie die Animationen sind nicht gerade flüssig zu nennen, enthalten aber sehr nett anzuschauende und vor allem witzige Effekte. Die Handlung dreht sich um einen kleinen, aber coolen Typen, der seine Spielsachen verloren hat. Als er die Hände (wohl aus Langeweile) tief in die Hosentaschen steckt, findet er sich plötzlich im Toyland wieder, in das ein paar wirklich üble Kerle seine Spielsachen entführt haben. Die Aufgabe unseres kleinen Helden besteht natürlich darin, den Burschen das Handwerk zu legen und all seine Spielsachen wieder zurückzuerobern. Doch das erweist sich als nicht ganz einfach. Wie bei einem Plattformspiel üblich, enthält auch Magic Pockets sowohl Elemente eines Strategiespieles (immerhin muß man sich gut überlegen, über welche Plattformen man ans Ziel eines jeden Levels gelangt) als auch Geschicklichkeitsprüfungen und natürlich Shoot'em'ups. Als Spiel im Spiel sind dann auch Dinge wie Fahrradrennen integriert.

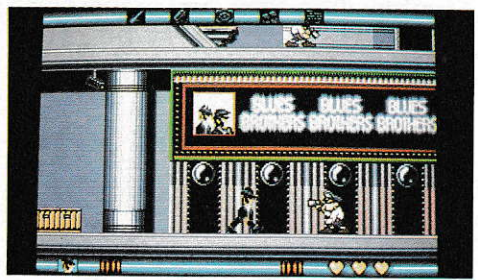
Die gegnerischen Sprites erscheinen in teilweise enormer Größe und verhalten sich recht unterschiedlich. Es gibt langsame Schnecken, fliegende Vögel und tollpatschige Roboter. In einigen Levels begegnet man auch gigantischen Schneebällen. Dabei sollte man tunlichst darauf achten, so viel wie möglich von den herumliegenden Süßigkeiten einzusammeln. Sie kommen der Gesundheit zugute (aber wohl kaum den Zähnen). Die Spielbarkeit ist als recht gut zu nennen, auch Neulinge in diesem Metier können das ein oder andere Level bequem schaffen. Die vielen kleinen grafischen Details (zum Beispiel beim Aufsammeln eines Kaugummis) machen das Spiel abwechslungsreich und witzig. Magic Pockets sollte bei keinem Anhänger von Plattformspielen in der Sammlung fehlen.

ddf/CM

BLUES BROTHERS

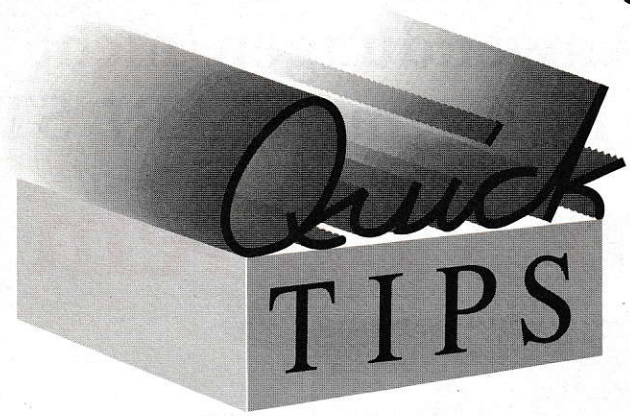


Falls Sie es mal gewagt haben, einem Musiker die Gitarre wegzunehmen, können Sie sich vorstellen, in welcher Laune die „Blues Brothers" jetzt sind. Den tingelnden Brüdern ist nicht nur die Klampfe gestohlen worden, sondern auch noch das Mikro, ein Verstärker, ein Konzertplakat und die Veranstaltungsgenehmigung. Der Dieb war ausgerechnet ein Polizist, einer von der schlimmen Sorte, versteht sich. Ihm sind die musizierenden Burschen ein Dorn im Auge. Diese Kanaille glaubt tatsächlich, daß sie die Jungs aus der Stadt vertreibt, wenn man ihnen jene Dinge klaut und versteckt, mit denen sie ihren Lebensunterhalt verdienen. Außerdem brauchen die Blues Brothers den ganzen Kram für das Konzert, das für denselben Abend überall angekündigt ist. In brüderlicher Eintracht machen sich Jake und Elwood auf den Weg, um die Gegenstände in der großen Stadt doch noch aufzuspüren. Im Zwei-Spieler-Modus macht es wohl am meisten Spaß, aber auch Einzelkämpfer werden gut unterhalten. Auf einer Orientierungsgrafik am Spielanfang zeigt der Computer, in welchem Haus die Suchexpedition beginnt. Darin spielt sich das erste Level ab, in dem ein einziger verlorener Gegenstand versteckt ist. Bevor man ihn nicht gefunden hat, gelangt man auch nicht in den nachfolgenden Spielabschnitt. In einigen Screens des ersten Levels finden sich Trittleitern oder Fahrkörbe zwischen den Ebenen. Darauf befinden sich hölzerne Kästen, Schallplatten, Hindernisse, Regenschirme, Sprungfedern und Betten. Komische Mischung, möchte man meinen. Aber jedes Ding hat seinen Zweck. Dazu muß man wissen, daß die suchenden Brüder nicht allein auf den Ebenen hin- und hereilen. Höchst eigenartige und gefährliche Zeitgenossen machen die Etagen unsicher: riesige Nagetiere, Sträflinge, Cops, Kampfköter und fliegende Bombenleger (Vögel!), um ein paar Beispiele zu nennen. Diesem Sammelsurium kuriosester Gegner sind die Kästen zugedacht. Jake oder Elwood schnappen sich die Holzbehälter und bombardieren damit die Angreifer - möglichst bevor diese mit Flinte, Reißzahn oder Bombe auf die beiden Musiker losgehen. Falls doch mal ein Geschoß heranpfeift, müssen sich die Brüder schnell ducken oder reißaus nehmen. Falls sie einen gegnerischen Treffer einstecken müssen, sieht der Spieler, wie die Energieanzeige langsam absinkt. Um sie wieder aufzufüllen, werden Schallplatten gesammelt. Jedes volle Hundert verleiht einen ordentlichen Schuß Mehrenergie. Hohe Hindernisse oder schlecht erreichbare Ecken bewältigen sie auf ihre Art: ein Sprung auf ein Bett oder auf eine Feder, und hui! fliegen sie im kühnen Bogen hoch oder drüber. Natürlich gelangt man damit nicht auf solche Ebenen hinauf oder hinunter, zu denen weder Lift noch Leiter führen. Wie bestellt, fliegen Ballons vorbei, die die kleinen Spielfiguren sicher nach oben tragen. Hinab geht es dann mit dem Regenschirm. Und schließlich erblickt man eine Fahne: das Level-Ende ist erreicht und hoffentlich das Instrument gefunden. Wenn ja, so blendet sich wieder das Bild von der Stadt ein und zeigt auf den nächsten Ort, der durchsucht wird.



Dabei muß der Spieler sich an die Reihenfolge halten, die der Rechner vorgibt. Das hat seinen Sinn. Als letztes soll die Konzertgenehmigung ausfindig gemacht werden, ohne die ja alle anderen Funde umsonst wären. Wenn das offizielle Erlaubnispapier aufgetaucht ist, findet das Konzert statt, und alles ist wieder so, wie das digitalisierte Bild vom Anfang zeigt: die „Blues Brothers" stehen auf der Bühne, ihre Augen glänzen vom Erfolg und vom Licht der Scheinwerfer. Solche Szenen kennen Kinogänger bereits aus dem gleichnamigen Kultfilm, da er bei dem vorgestellten „offiziellen Computerspiel zum Film" als Vorlage diente. Hersteller ist Titus Software, und ihnen gebührt Lob für die hübsche und witzige Umsetzung in das kurzweilige Plattformspiel. Die Titus-Leute zauberten mit ihren digitalen Farbtöpfen und schufen eine fröhliche Comic-Atmosphäre. Sie ließen ihre Phantasie spielen und ertüftelten für jedes Level eine andere Umgebung, die sie detailliert und niedlich malten. Sogar Einzelheiten in den Gesichtern sind zu erkennen, beispielsweise bei dem Grantelhuber mit der Aktentasche, bei der Bilderbuchoma im Einkaufswagen und selbst beim großmäuligen Comic-Hund. Alles wird durch eine gute Spielbarkeit gekrönt und durch ein hübsches, deutsches Handbuch dokumentiert. Außerdem noch etwas, was bei Computerspielen eher unüblich ist: die Bluesbrüder sind kooperativ, und folglich arbeiten hier nicht zwei Spieler gegeneinander, sondern sammeln und kämpfen miteinander. Wer welchen Gegenstand findet, ist der Software egal. „Blues Brothers" ist eben ein Spiel für teamfähige Leute.

CBO



TT-Auflösungen in PURE C?

Beim Einsatz des bei PURE C mitgelieferten Borland-Grafik-Interfaces (BGI) wird jeder Besitzer eines TT wohl schon festgestellt haben, daß in der Header-Datei des BGI namens „GRAPHICS.H" die Auflösungen des TT nicht berücksichtigt sind. Wie stellt man nun bei einem Programm, das in einer der TT-Bildschirmmodi laufen soll, die richtige Auflösungsstufe ein?

Die Lösung des Rätsels ist denkbar einfach, man baut in den PURE-C-Code folgendes Handling ein:

```
int treiber = DETECT,
        aufloesung = STHIGH;
main (...)
{
...
initgraph
(&treiber,&auflösung,"");
...
}
```

Beim ersten Starten des hernach compilierten Programms taucht schließlich folgende Fehlermeldung auf:

```
„Ungültiges Handle!
Handle = $FFFF
|weiter| |tolerieren|"
```

Wenn man den Button „tolerieren" durch Return bestätigt, wird das Programm ganz normal fortgesetzt.

Auch die Befehle *getmaxx* und *getmaxy* liefern die korrekten Werte (z.B. 640 * 480 für VGA). Bei jedem weiteren Start erscheint nun die Handle-Fehlermeldung nicht mehr, solange die CPU kein Reset erfährt oder der Rechner neu bootet.

M. Lupp, Bexbach

Haben auch Sie einen *Quick-Tip* ?

Standen Sie auch einmal vor einem kleinen, aber schier unlösbarem Problem? Dann, durch Zufall bekamen Sie einen Tip und schon war es gelöst.

Ähnlich haben wir uns diese neue Rubrik in der ST Computer vorgestellt. Aufgerufen sind auch Sie, liebe Leser(innen)! Geben Sie Ihre Erfahrungen weiter, egal, ob es um Anwendungen, Programmieren o.ä. geht.

Wir sammeln Ihre (und unsere) Tips und stellen Sie ggf. in den Quick-Tips vor.

Einsendungen an:

MAXON Computer
ST Computer Redaktion
Stichwort: Quick-Tip
Industriestr. 26
D-6236 Eschborn

Mehrmaliger Aufruf des Omikron-Interpreters

Wer hat diese Situation nicht schon einmal erlebt! Da schreibt man gerade an einem neuen Programm und benötigt aus einem alten eine Prozedur.

Da man ja ein unordentlicher Mensch ist und sich keine Blockbibliothek eingerichtet hat, wird man wohl notgedrungen folgendermaßen vorgehen.

1) neues Programm sichern
2) altes Programm laden
3) Prozedur suchen und als Block abspeichern
4) neues Programm wieder laden
5) Block in das neue Programm einfügen

Folge der ganzen Hin-und-her-Laderei: Ehemals gesetzte Markierungen sind futsch. Gerade bei längeren Programmen kann dies sehr ärgerlich sein. Aber da gibt es ja noch den Menüpunkt EXEC.PRG! Warum nicht einfach Omikron.BASIC nochmals starten und schauen, ob dies funktioniert? Und siehe da: es funktioniert!

1) neues Programm nicht sichern, sondern Omikron.BASIC (2) über EXEC.PRG nochmals starten.
2) altes Programm laden
3) Prozedur suchen und als Block abspeichern
4) Omikron.BASIC (2) verlassen
5) wir befinden uns wieder in Omikron.BASIC (1). Und siehe da: alle Markierungen sind noch gesetzt.

Also noch eben schnell den Block eingefügt: Fertig!

Ob diese Vorgehensweise Nicht-Festplattenbesitzern zu empfehlen ist, ist fraglich. Doch gerade wenn man im Besitz einer Harddisk ist (und wer ist das heute nicht?), kann man sich auf diese Art und Weise eine Menge Zeit und Arbeit sparen.

D. Hagedorn, Paderborn

Schriftenwechseldich

Finden Sie nicht auch, daß die GEM-Systemoberfläche in unserem ATARI-Rechner etwas öde ist? Besonders in Richtung Systemschrift (System-Font) habe ich da eine Idee für eine Abhilfe. Mit meinem kleinen GFA-BASIC-Programm läßt sich sehr einfach die Systemschrift ändern. Wir müssen zuerst ein Control-Array besetzen, dann den Aufruf für dessen Änderung, anschließend (das Wichtigste!) die Effekt-Bits setzen und zum Schluß noch die Änderung aktivieren.

A. Hitzschke, Papenburg

```
' GEM-Schrift ändern
' GFA-BASIC ab V:1.0
' Andre` Hitzschke
'
PRINT "~~~ Schriftartänderer ~~~"
PRINT "  von Andre` Hitschke"
PRINT
PRINT
PRINT " [ 0] - normal"
PRINT " [ 1] - fett"
PRINT " [ 2] - hell"
PRINT " [ 4] - kursiv"
PRINT " [ 8] - unterstrichen"
PRINT " [16] - hohl"
PRINT
'
INPUT "Schriftart: ",a
'
DPOKE CONTROL+2,0      !Control-Array besetzen
DPOKE CONTROL+6,1      !Aufruf zum Ändern
DPOKE CONTROL+12,1
DPOKE INTIN,a          !Effekt-Bits setzen
VDISYS 106             !Schrift aktivieren
```

Einfache Zeitaddition

```
REPEAT
  CLS
  Ges_Min=0:Ges_Sek=0
  PRINT " kleine Zeitaddition"; TAB (56);
  "zweimal RETURN für Ende"
  PRINT " "+"-"*78
  '
  REPEAT
    PRINT @(4,1);"Eingabe: __ min __ sek"
    Minuten$="":Sekunden$=""
    INPUT @(4,10);Minuten$ USING "0"+">",,
    2:Minuten= VAL(Minuten$)
    INPUT @(4,18);Sekunden$ USING "0"+">",,
    2:Sekunden= VAL(Sekunden$)
    Summiere_Zeit(Minuten,Sekunden,Gesamt_Zeit$)
    PRINT @(4,40);"Gesamtzeit: ";Gesamt_Zeit$
  UNTIL Minuten=0 AND Sekunden=0
  '
  INPUT @(7,1);"Nochmal (j/n)? ";Jn$ USING "1+j+
  n"+">",,1
UNTIL Jn$<>"j"
END
'
DEF PROC Summiere_Zeit(In1,In2,R Out$)
    Ges_Min=Ges_Min+In1
    Ges_Sek=Ges_Sek+In2
    IF Ges_Sek>59 THEN Ges_Min=Ges_Min+1:Ges_Sek=
    Ges_Sek-60
    USING "### min ## sek"
    Out$= STR$(Ges_Min*100+Ges_Sek)
    USING
RETURN
```

Wer seine gesamten Schallplatten und CDs in einer Datenbank archiviert, gibt gewöhnlich auch ein Feld für die Gesamtlaufzeit an. Leider sind die Gesamtlaufzeiten in den seltensten Fällen auch angegeben. Also heißt es, mühsam die Laufzeiten der einzelnen Stücke per Hand zu addieren.

Einfacher geht es mit folgendem kleinen Programm, das natürlich noch beliebig erweiterbar ist (z.B. Abfangen von unsinnigen Eingaben etc.). Das Programmchen wurde mit Omikron.BASIC geschrieben. Es sollte aber selbst für den puren Anfänger keine Schwierigkeit sein, es in andere Sprachen zu übersetzen.

Tip: Compilieren Sie das Programm als TOS-Applikation. Dann steht es Ihnen auch in Datenbanken, die einen Programmaufruf erlauben (z.B. Phoenix/ASH), ständig zur Verfügung. Wer will, kann aufgrund dieser Idee ja auch sein eigenes Accessory zurechtbasteln.

D. Hagedorn, Paderborn

Noch eine Boot-Hilfe

Obwohl mittlerweile schon eine ganze Reihe von Vorschlägen veröffentlicht wurde, die das Ziel haben, den ST beim Booten von der Festplatte zu etwas mehr Geduld zu veranlassen, sei hier noch eine weitere Variante vorgestellt, die einige Vorzüge in sich vereinigt.

Das Funktionsprinzip ist einfach, aber wirkungsvoll: Bei Einschalten des Rechners wird der Monoflop gesetzt und hält für eine einstellbare Zeit (ca. 5-30 Sek.) den Eingang der Reset-Schaltung auf Low, so daß der Boot-Vorgang mit entsprechender Verzögerung startet; wird dagegen der Reset-Knopf gedrückt, bleibt die Zusatzschaltung inaktiv, und der Boot-Vorgang startet ohne unnötige Verzögerung.

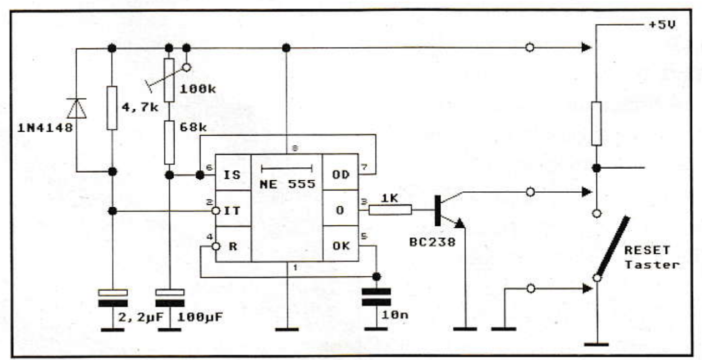
Diese Eigenschaft besitzen jedoch auch andere, bereits veröffentlichte Schaltungen, die die Betriebsspannung(en) bzw. die Netzspannung verzögert zuschalten; die hier vorgestellte Lösung hat gegenüber diesen jedoch vor allem zwei Vorteile:

1. Der schaltungstechnische und damit auch der finanzielle Aufwand sind geringer (kein Relais erforderlich).
2. Es sind nur minimale Eingriffe in den Rechner (es müssen keinerlei Leitungen aufgetrennt werden) bzw. keine Arbeiten auf der Netzspannungsseite, die für den Laien doch Risiken bergen, erforderlich.

Die Schaltung findet problemlos auf einem ca. 3x3cm großen Stück Universalleiterplatte Platz. Mit einer Leuchtdiode und einem Widerstand zwischen +5V und dem Kollektor des Transistors kann man die Funktion vor dem Einbau überprüfen und auch schon die Zeitverzögerung einstellen. Der Einbau schließlich ist denkbar einfach: die drei Anschlußpunkte findet man auf der Platine alle in unmittelbarer Nähe des Reset-Tasters. Auf keinen Fall sollte man die Schaltung an den Reset-Anschluß des Prozessors anschließen!

Bleibt noch zu erwähnen, daß die vorgestellte Schaltung bereits in mehreren Rechnern vom 260er bis zum Mega ST eingebaut wurde und in allen seit Monaten zur vollsten Zufriedenheit funktioniert.

W. Schneider, Berlin



Ärger über den Boot-Umschalter

Eigentlich war ich begeistert, als ich den Quicktip von Herrn Brünjes in Heft 10/91, Seite 182, las und machte mich sogleich mit dem enpfohlenen Teppichmesser ans Werk. Erst drückte ich vorsichtig, dann mit etwas mehr Kraft und dann „mit einer Portion Kraft“ (Zitat), aber nichts passierte, außer daß die Klinge stumpf wurde. Schließlich hatte ich die Nase voll und setzte noch mehr Kraft ein - das Resultat: ein komplett zerfetztes Teppichmesser, einige unterbrochene Leiterbahnen und tiefe Kratzer im Soundchip.

Nach siebenstündiger Arbeit hatte ich es endlich geschafft aus dem Schrott wieder einen funktionstüchtigen Rechner zu basteln. Deshalb meine Fragen: Warum untertreiben Sie so sanft bei einer Umbauanleitung, und wer ersetzt mir Material, Zeit und Aufwand?

P.-M. Jung, Kelkheim/Taunus

Antwort von Herrn Brünjes:
Um ganz offen zu sein: Seit dem Umbau meines STE in dem zitierten Quicktip habe ich noch eine Reihe anderer STs mit dem Umschalter versehen und hatte dabei NIE Probleme. Wie Herr Jung letzten Endes vorging, und was er dabei falsch machte, kann ich leider aus der Distanz nicht nachvollziehen.

Das Durchtrennen der Pins kann natürlich auch mit einem feinen Elektroseitenschneider erfolgen, so erspart man sich eventuelle Ausrutscher und dadurch zerschnittene Leiterbahnen. Für Leser, die den Umbau jedoch 100%ig durchführen wollen, gibt es die Möglichkeit, den Soundchip ganz aus der Grundplatine auszulöten und dann die Pins 19 und 20 nach oben zu biegen. Anschließend muß der Chip nur wieder eingelötet werden. ACHTUNG! Diese Methode sollten nur Leser anwenden, die reichlich Erfahrung im Umgang mit ICs haben und über das richtige Werkzeug verfügen!

M. Brünjes, Bremen

Anmerkung der Redaktion:
Weil Herr Jung die Frage der Haftung anschnitt, möchte ich sehr gerne auf unser Impressum verweisen, in dem (ganz unten) ein Haftungsausschluß nachzulesen ist. Nur bei nachweisbarem Vorsatz und bei einem Beschreibungsfehler müßte der Autor selbst einen Schadensersatz übernehmen.

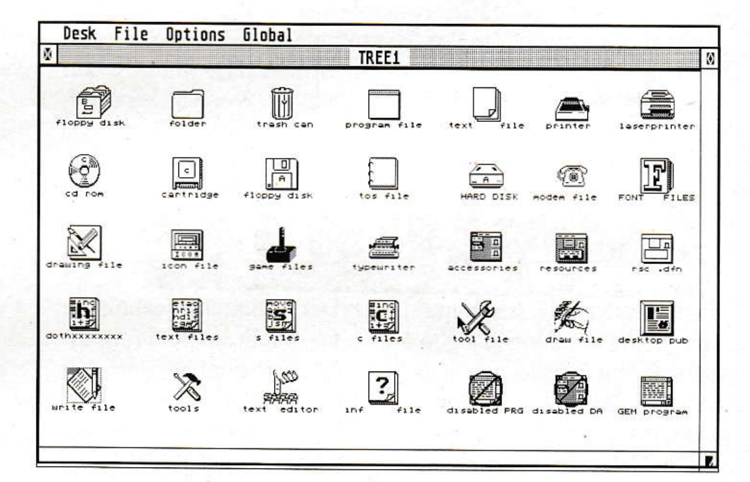
Die Redaktion kann nicht abschätzen, inwieweit welche Bastelanleitungen den Fähigkeiten bestimmter Leser entsprechen. Solche Beschreibungen in die Tat umzusetzen, unterliegt immer der Verantwortung des Ausführenden.

DK

Eigene Desktop-Icons unter TOS 2.05

Um sich eigene Icons für das neue Desktop der MEGA-STEs und TTs zu erstellen, benötigen Sie lediglich zwei Disketten aus der Sonderdisk-Serie dieser Zeitschrift. Dies sind das RCS 1.4 (SD 02) und das Programm IconDesign (SD 17). Gehen Sie bei der Erstellung der Icons folgendermaßen vor:

1) Entwerfen Sie die Icons mit IconDesign, wobei Sie die Box auf x=64 und y=32 einstellen und den Ausgabe-Schalter auf RSC-Ausgabe stellen.
2) Erstellen Sie ein Backup der Datei DESKICON.-RSC für den Fall, daß doch mal etwas schiefgehen sollte.
3) Öffnen Sie im RCS die Datei DESKICON.RSC. Daß das RCS kein Definitions-File findet, ist in diesem Falle egal. Nun sehen Sie genau einen Objektbaum vor sich. Ändern Sie den Typ des Objektbaums in FREE und öffnen Sie ihn. Dann sollten Sie die gesamten Desktop-Icons vor sich sehen.
4) Um nun Ihre erstellten Icons zu benutzen, klicken Sie einfach ein unbenutztes Icon an, und laden Sie die Daten und die Maske Ihrer Icons.
5) Speichern Sie das geänderte RSC-File unter DESKICON.RSC und kopieren Sie es auf Ihre Bootpartition bzw. Diskette.



Viel Spaß beim Entwurf Ihres neuen Desktops!

D. Hagedorn, Paderborn

Zur Megamax Modula-2 Bibliothek

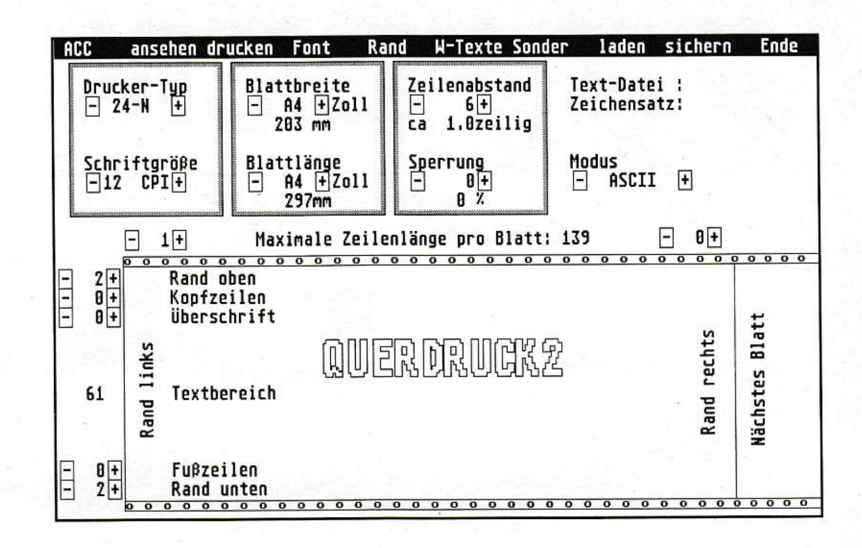
Die Funktion *Directory.FileExists (REF fileNAme: ARRAY OF CHAR)* der Megamax Modula-2-Bibliothek arbeitet fehlerhaft! Erfragt man mit ihr die Existenz eines Files, während eine beliebige Datei geöffnet ist, sind anschließend keine Zugriffe mehr auf diese Datei möglich. Stattdessen liefert *Files.State* undefinierte Feldnummern. Eine zu *FileExists* äquivalente Funktion kann jedoch schnell selbst programmiert werden. Die benötigten Funktionen aus *Files* dürften ohnehin von jedem Programm, das mit Dateien operiert, importiert werden, so daß sich die Funktion problemlos in jeden Quelltext einbauen läßt.

B. Stratmann, Recklinghausen

```
PROCEDURE FileExists (REF FileName: ARRAY OF
CHAR): BOOLEAN;
(* liefert TRUE, wenn eine DAtei namens
"FileName" existiert *)

VAR
   f:     Files.File;
   ok:    BOOLEAN;
BEGIN
   Files.Open(f,FileName,readOnly);
(* nur, wenn "FileName" bereits existiert *)
   ok:=(Files.State(f)=MOSGlobals.fOK);
   Files.Close(f);
   Files.ResetState(f);(* vorsichtshalber *)
   RETURN ok;
END FileExists;
```

Kreuz und quer



Da haben Sie nun auf kompetenten Rat einer ST-COMPUTER-Zeitschrift das neue Datenbankprogramm gekauft und sehen mit Schrecken, daß die wunderschön zusammengestellte Liste um nichts auf der Welt auf ein Blatt Papier passen will.

Gerade bei umfangreichen Listen, die uns hauptsächlich von Datenbank- oder Tabellenkalulationsprogrammen überreicht werden, tut Überblick not, wenn in der Breite nicht alles auf das Papier gedruckt ist. Selbst die ausgefeiltesten und kostspieligsten Programme lassen uns Anwender mit dem Druckergebnis alleine, denn sie berücksichtigen oftmals nur die normale DIN-A4-Breite, und das war's dann auch. Dann hilft nur noch eine große Portion Hoffnung, daß uns die Riesenliste nicht zu sehr zerstükkelt aus dem Drucker kommt, und eine ebenso große Portion Klebestreifen, um die Liste wieder zurechtzukleben.

Von vielen unbeachtet, gibt es seit 1988 schon einen cleveren Ausweg aus dem Dilemma; das Programm heißt QUERDRUCK, wir stellten es in Heft 9/90 der Zeitschrift ST-COMPUTER schon einmal kurz vor. Mittlerweile hat sich in der Entwicklung dieses Programms einiges getan, das die Bezeichnung QUERDRUCK 2 rechtfertigt und für uns Grund genug war, diese Lösung für Ausdrucke in „Überbreite" noch einmal in Augenschein zu nehmen.

Die erste Version von QUERDRUCK konnte schon Texte oder andere Druckvorlagen bis zu einer Breite von 32000 Zeichen auf fast allen 9- bzw. 24-Nadeldruckern ausgeben. Anmerkung aus dem Handbuch: „Falls Sie es übers Herz bringen, die volle zulässige Breite von 32000 Zeichen zu drucken, dann sind Sie reif für das Buch der Rekorde." Der Trick an der Geschichte war schon damals, den Text einfach um 90° zu drehen. Das machte es natürlich notwendig, daß der auszugebende Text intern umgesetzt (in ASCII konvertiert) und mit einem eigenem Zeichensatz zu Papier gebracht würde.

Die 2. Generation

Aufgrund zahlreicher Kundenrückmeldungen kann die 2. Version von QUERDRUCK heute noch wesentlich mehr. Mußten früher alle auszudruckenden Texte noch im ASCII-Format angeliefert werden, finden heute drei „Wordplus-Modi" Anwendung. So bleiben u.a. wichtige Attribute aus einer DOC-Datei erhalten, wie im *WP-Modus 1* Fettschrift, Unterstreichung und Leerzeichen (z.B. wegen Blocksatz). Mit *WP-Modus 2* kann man die Übernahme sogenannter Kopf- und Fußzeilen bestimmen und schließlich im Modus 3 die gesamten Seitenformateinstellungen (also auch die harten Seitenumbrüche usw.) verwenden.

Natürlich möchte QUERDRUCK 2 nicht nur Textdateien zur Bearbeitung haben, sondern versteht sich mit einer großen Anzahl bekannter Anwenderprogramme auch sehr gut. Auch hier haben lebhafte Kundenmeldungen aus der 1. Version dazu geführt, daß Unverträglichkeiten abgestellt werden konnten. Um alle denkbaren Anwendungsfälle abzudecken, besteht die Möglichkeit QUERDRUCK 2 alternativ als Accessory zu laden. Es braucht z.Zt. aber mit 200 KByte noch relativ viel Arbeitsspeicher, wodurch der Hersteller selbst einen Rechner mit mindestens 2 MByte Arbeitsspeicher empfiehlt. Also Achtung: Die Bequemlichkeit (als ACC) kostet auch hier ihren Preis (RAM-Arbeitsspeicher)!

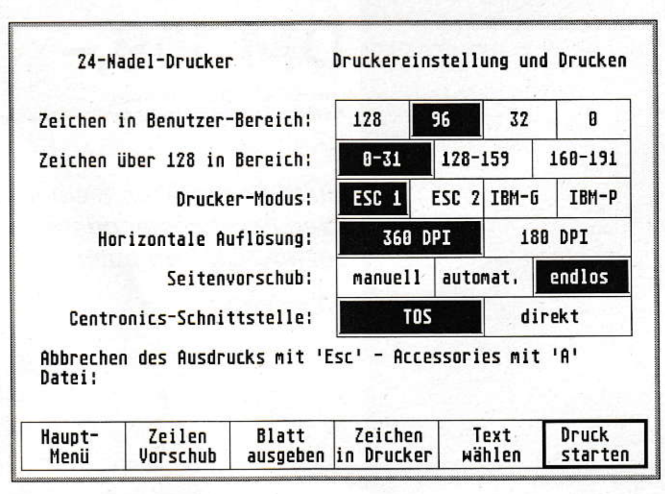


Bild 2: So präsentiert sich das Menü zur Druckermodifikation.

Gleichgültig, wie das Programm später zur Arbeit gerufen wird, ein Prinzip der Vorgehensweise hat sich nicht geändert: die auszudruckende Datei (gleichgültig von welchem Anwenderprogramm sie auch stammt) muß immer abgespeichert vorliegen. Wer nun an die direkte Übernahme eines Textes aus dem Arbeitsspeicher in das QUERDRUCK-ACC denkt oder gar an eine Einbeziehung des GEM-Klemmbrettes glaubt, sieht sich leider alleingelassen. (Aber es muß ja noch Ideen für eine 3. Version geben.)

Auf gute Zusammenarbeit

So gesehen ist also die Zusammenarbeit mit Anwenderprogrammen auf den Zugriff (des QUERDRUCK-Programms) auf die Datendateien beschränkt. Grundsätzlich ist also bei Textverarbeitungen darauf zu achten, daß sie zumindest „Wordplus"-Dateien erzeugen, sonst hilft es nichts, und man muß sich mit reinen ASCII-Dateien begnügen. Also ist die Datenübernahme bei dem Programm WORDPLUS am unproblematischsten (zumal dann in QUERDRUCK 3 verschiedene Modi zur Verfügung stehen - siehe oben). Weiterhin relativ unproblematisch präsentiert REDAKTEUR seine Textdaten, weil es auch das Wordplus-Format erzeugen kann. Bei THAT'S WRITE ist der reine ASCII-Text erst dann brauchbar, wenn vor dem Abspeichern die Option „CR + LF nach jeder Zeile" aktiv ist. Besonderheit: Tabulatoren aus THAT'S WRITE kann QUERDRUCK mitverwenden.

SIGNUM und SIGNUM 2 machen es dem Text etwas schwerer, weil sie grundsätzlich das ASCII-Zeichen Nr. 12 für einen Seitenvorschub verwenden. Wenn die Seitenlänge aber auf maximal groß gestellt wird, kann in QUERDRUCK sogar der WP-Modus 1 (Schriftattribute übernehmen) Verwendung finden.

Die Schwierigkeiten dieser und anderer Textprogramme liegen also hauptsächlich in der unterschiedlichen Bezeichnung der Befehle für den Seitenaufbau und die Formatierung. Ist ja eigentlich auch einleuchtend, daß die Textprogramme mit Vorliebe ihre eigenen Druckertreiber bedienen wollen und dafür die exotischsten Steuersequenzen verwenden.

Aus Tabelle und Datenbank

Wahrscheinlich wird es in der späteren Arbeit mit QUERDRCUK 2 nicht allzu oft vorkommen, einen Text mit einer immens großen Zeilenbreite ausdrucken zu lassen (denken Sie vielleicht an das Buch der Rekorde?). Die interessantere und auch umfangreichere Zielgruppe bilden sicher all jene Anwender, die ein vernünftig aufgebautes Listenbild aus Tabelle und Datenbank wünschen. Also dürfte die sinnvolle Zusammenarbeit mit solchen Programmen bei einer Beurteilung eher im Vordergrund stehen.

Die Angelegenheit gestaltet sich dann aber auch ein kleinwenig komplizierter; betrachten wir die zwei möglichen Extreme: Da gibt es einerseits entsprechende Anwenderprogramme, die die Seitengestaltung völlig selbst übernehmen, da muß QUERDRUCK 2 sich darauf einstellen. D.h. man muß schon im vorhinein wissen, wieviele Zeilen später quer auf dem Ausdruck erscheinen sollen. Andererseits gibt es Anwenderprogramme, die eine Textdatei erzeugen, mit diesen hat QUERDRUCK 2 ein leichtes Spiel, denn es kann seine individuellen Einstellungen später vornehmen.

```
Die eingestellte Blattlänge ist größer        weiter  1
als vom Laser erlaubt.                        zurück  2
Nicht tragisch, kann gewollt sein.

(Kassette: 11.2, Einzelblatt 10.6 Zoll)
```

```
Die eingestellte Blattbreite ist größer       zurück  1
als vom Laser erlaubt.      ==>Chaos

(Kassette: 7.7, Einzelblatt 8 Zoll)
```

```
Möchten Sie die Seite 2     drucken                1
                            überspringen           2
oder umschalten auf:        gesamten Text drucken  3
```

Bild 3: Ausführliche Warnboxen weisen auf Einstellfehler hin.

Beispiele: In BASICALC bzw. BASICHART legt man Zeilen- und Seitenlänge unabhängig von den wirklichen Papierdimensionen fest, halt eben den gesamten druckbaren Bereich. Es wird eine reine ASCII-Datei angelegt. Ähnlich ist die Prozedur bei VIP-professional, LDW-Powercalc. K-SPREAD 3 benötigt verschiedene Grundeinstellungen, die auf der QUERDRUCK 2-Diskette mitgeliefert werden.

Die Oberfläche

Also gleichgültig, wie QUERDRUCK 2 zur Arbeit gerufen wird (entweder als PRG oder als ACC), präsentiert es sich uns als eine riesige Dialogbox. Will heißen: Eigentlich ist es gar kein reinrassiges GEM-Programm, weil die vermeintliche Menüleiste keine Pulldown-Menüs verbirgt, sondern wie normale Dialogknöpfe (Buttons) funktioniert.

Den größten Teil dieser Oberfläche nimmt (rechts unten) symbolisch ein querliegendes Endlosblatt ein, aus dem man alle Formateinstellungen ablesen kann. Mit der Funktion „ansehen" kann der spätere Drucktext zur Kontrolle auch dorthin projiziert werden.

Die Querdruckvorlage sieht Einstellungen für linken und rechten sowie für oberen und unteren Rand mit evtl. gewünschten Kopf- und Fußzeilen vor. Gleichzeitig errechnet das Programm daraus die maximale Zeilenlänge sowie die Zeilenzahl pro Blatt, genannt: Textbereich. Andere Parameter, in den (links oben zu sehenden) drei Kästchen beeinflussen ebenfalls die Größe des bedruckbaren Bereiches, als da wären: Schriftgröße (in Zeichen pro Zoll), die physikalische Blattbreite und -länge sowie der verwendete Zeilenabstand und evtl. eine Sperrung (variabler Wortzwischenraum) im Text. Alle Einstellungen sind mit den „+"- und „-"-Knöpfen veränderbar.

Der Text kann kommen

Es ist sehr zu empfehlen, nach diesen Grundeinstellungen (wozu natürlich auch der Druckertyp gehört) alle Parameter zu sichern. Danach kann der Text kommen. Der Dialog-Button „laden" bringt uns die Dateiauswahlbox auf den Schirm, wodurch hernach die Textdatei in den Arbeitsspeicher gelangt. Wie schon erwähnt kann der selektierte Text auch auf dem Bildschirm in dem Querdruckmuster sichtbar gemacht werden, um sich einen genauen Eindruck vom späteren Ausdruck zu verschaffen. Weil dies erfahrungsgemäß nur in der Eingewöhnungsphase oder beim Kontakt mit einem neuen Anwenderprogramm vonnöten ist, kann man sich dieses Ansehen auch getrost sparen.

Wie schon weiter oben erwähnt, gestaltet QUERDRUCK 2 den Text völlig um, weshalb es ihn vorzugsweise als ASCII-Vorlage wünscht. Nur bei einigen Pro-

grammen können Schriftattribute mit übernommen werden und erscheinen stilecht auch im Querausdruck. Dennoch ist ein Arbeitsprinzip ganz grundsätzlich, und zwar die Umsetzung von ASCII (von möglichen, erlaubten Attributen isoliert) in einen eigenen QUERDRUCK-Zeichensatz. Dazu ist es nötig, einen der mitgelieferten Zeichensätze zu wählen. Es sind z.Zt. je 2 Zeichensätze für jeden Druckertyp auf der Diskette zu finden, und zwar einmal „dünn" und einmal „fett". Zur Eigenkreation von Zeichensätzen kommen wir gleich noch zurück.

Schwarz auf weiß

Den Lohn aller Mühen (und des Programmkaufs) erhält man schließlich, wenn der Drucker die überbreiten Vorlagen wunschgerecht zu Papier bringt. Im Fortgang der Arbeitsweise sollte der Menüpunkt „drukken" gewählt werden, was uns je nach Druckertyp eine andere, weitere Dialogbox offenbart. In dieser Dialogbox (siehe Bild 2) sind nunmehr alle spezifischen Einstellungen des Druckers gefragt. QUERDRUCK 2 unterstützt mittlerweile alle bekannten 9- und 24-Nadler und die ATARI-Laser SLM 804 und SLM 605. Auch diese typrelevanten Parameter werden mit den formatrelevanten gemeinsam abgespeichert und müssen in aller Regel nur einmal vorgenommen werden.

Jetzt wäre eigentlich die Besprechung dieses unscheinbaren, aber höchst nützlichen Programms zu Ende, wenn es da nicht noch einige Besonderheiten gäbe:

Randzeichen

Leider bekam der ATARI-Rechner von seinen Urvätern eine ganze Anzahl in unseren Breiten völlig unnützer Schriftzeichen ins ROM gelegt (hebräisch und griechisch). Andere Rechner, die im großen und ganzen nicht unbedingt so fortschrittlich und komfortabel sind (unvermeidliche versteckte Anspielung des Autors auf den Uraltstandard MS-DOS), zeigen sich da aufgeschlossener und bekamen einen eingeschränkten Grafikzeichensatz (IBM #2) mitgeliefert.

Um diese Grafikzeichen dennoch nutzbar zu machen, ist in QUERDRUCK 2 die Option „Rand" vorgesehen. Es öffnet sich daraufhin eine Übersicht der erweiterten ASCII-Tabelle (von Zeichen Nr. 160 bis 255), in denen die Grafiksymbole zu finden sind. Mit einfachem Anklicken der Symbole werden diese für den späteren Ausdruck gewählt. Ab diesem Zeitpunkt zaubert das Programm die passenden Randlinien um den Text bzw. die Zahlenkolonnen herum.

In diesem Zusammenhang macht es sich sehr sinnvoll, vor dem eigentlichen Ausdruck noch einmal eine vollständige ASCII-Tabelle anzeigen zu lassen, welche Zeichen denn nun überhaupt verfügbar sind.

Kopf und Fuß

Gerade bei größeren Vordrucken, die zudem über mehrere Seiten gehen, kann es sehr sinnvoll sein, die einzelnen Blätter zu „betiteln". Dafür sind die Positionen für Kopf- und Fußzeilen vorgesehen. Dabei handelt es sich um Wiederholtexte und weitere Angaben (Seitennummer, Seitenanzahl, Datum, Uhrzeit und Dateiname), die wahlweise am rechten oder linken Rand bzw. mittig auf dem Querformat positioniert werden. Es gibt sogar Modi (siehe oben), die diese Wiederholtexte aus der vorher benutzten Textverarbeitung übernehmen können.

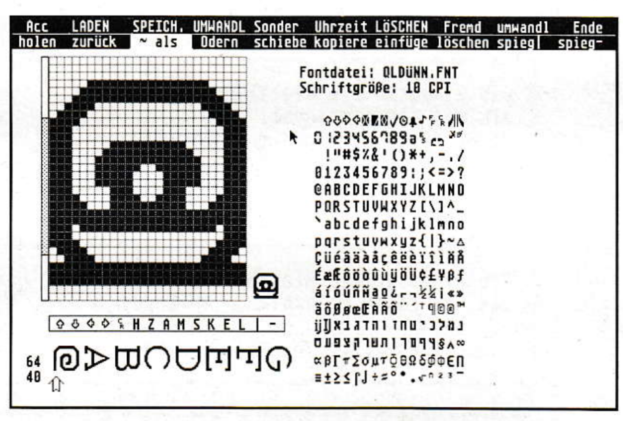


Bild 4: Der eigene Zeichensatzeditor Q2FONT

Mitteilungen

Weil natürlich niemand vor Bedienungsfehlern gefeit ist, sind in dem Programm aussagekräftige Fehlermeldungen eingebaut. Gerade wenn die Formateinstellungen nicht mit der tatsächlichen Papiergröße übereinstimmen, kann es zu unerwünschten Druckergebnissen kommen. QUERDRUCK 2 kann nun feststellen, daß z.B. die eingestellte Blattlänge und -breite mit dem eingestellten Drucker nicht zu verwirklichen ist (siehe Bild 3). Erst wenn alle Angaben aufeinander passen, erfolgt widerspruchslos der lang ersehnte Ausdruck.

Neuer Font gefällig?

Es hat sich gezeigt, daß beim ACC-Einsatz mit einem Laserdrucker Speicherplatzprobleme auftauchen. QUERDRUCK 2 reserviert (außer seinem eigenen RAM-Platz) noch etwa 110 KByte für den benutzten Zeichensatz. Besonders wenn Programme zusätzlich noch im Arbeitsspeicher lauern (erst dann macht ein ACC-Einsatz ja Sinn), ist der freie RAM-Speicher schnell aufbebraucht. Man bedenke: Eine Druckseite für den Laser nimmt selbst schon 1 MByte in Anspruch. Eine Abhilfe kann dann sein, den verwendeten Zeichensatz zu verringern. Diese und noch andere Aufgaben hat der eigene Zeichensatzeditor „Q2FONT" (siehe Bild 4).

Als zweite Anwendung des Q2FONT ist das Ändern bestehender Zeichen denkbar. Wenn Ihnen also der Schnörkel beim Klammeraffen oder die Dicke des i-Punktes nicht gefällt - alles kein Problem. Natürlich kann man sich einen eigenen, völlig neuen Zeichensatz mit Q2FONT zusammenbasteln, z.B. aus einem 9-Nadler-Font einen für 24-Nadler fertigen.

In dem Zeichensatzeditor erscheint eine zweizeilige Pseudomenüleiste (die auch wieder nur Knöpfe und keine Menüs enthält), die alle Arbeitswerkzeuge zur Manipulation bestehender oder neuer Schriftzeichen bereithält. Links im Bild erkennt man das Pixel-Raster mit einem gewählten Zeichen, in dem per Mausklick einzelne Pixel ein- oder ausgeblendet werden können. Im rechten Bildteil ist die verwendete ASCII-Tabelle komplett zu sehen. Leider funktioniert die Auswahl des nächsten Zeichens hier nicht per Mauspfeil und Doppelklick direkt, man muß erst den Menüpunkt „holen" aktivieren. Ich habe das aus anderen Zeichensatzeditoren etwas komfortabler in Erinnerung. Beispielsweise das Löschen von Zeichen aus der Tabelle funktioniert einfacher: Anklicken (auch mehrere) und auf Menüpunkt „löschen" gehen.

Unter dem Editier-Rasterfeld ist eine Symbolleiste zu erkennen, die neben einigen Schiebefunktionen die Optionen der zweiten Menüleiste noch einmal wiederholt (und genauso funktioniert - warum denn bloß?). Weiter darunter ist eine „Hilfszeile" untergebracht. Sie soll acht der benachbarten Zeichen (vom gewählten ausgehend) noch einmal etwas größer darstellen, um einen Eindruck über die Dimensionen und das Aussehen zu erhalten - warum auch immer. Bei einigen Schriftgrößen ist die Darstellung etwas verzerrt, weil es sich hier ja nicht um den sogenannten Bildschirm-Font, sondern schon um den Drucker-Font handelt.

Sehr schön ist nun aber die Möglichkeit, Zeichensätze aus einem beliebigen Bild im Screen-Format (640 * 400 Pixel, 32

KByte Datei) ausschneiden zu können. Lieferanten fremder Zeichensätze werden das nicht unbedingt mit Freude vernehmen, auch gilt es, das Urheberrecht für Zeichensätze zu beachten.

Ein Schlußwort

QUERDRUCK 2 ist ein äußerst nützliches Programm für all jene, die über den Blattrand hinausblicken wollen. Immer dann, wenn das Tabellenprogramm überbreite Listen liefert (was nicht unbedingt selten vorkommt), ist QUERDRUCK 2 gefordert. Wie schon erwähnt, lassen ausnahmslos alle Anwenderprogramme ihre Nutzer dann im Regen stehen, wenn das Papier breiter als DIN A4 bedruckt werden soll. Das, was sich wie eine ernstgemeinte Schelte an die Druckertreiberprogrammierer liest, ist auch gleichzeitig die Rechtfertigung, warum es ein Programm wie QUERDRUCK 2 geben muß. So gesehen hat QUERDRUCK 2 seinen Platz überall dort, wo man mehr Breite in seinen Ausdrucken haben möchte.

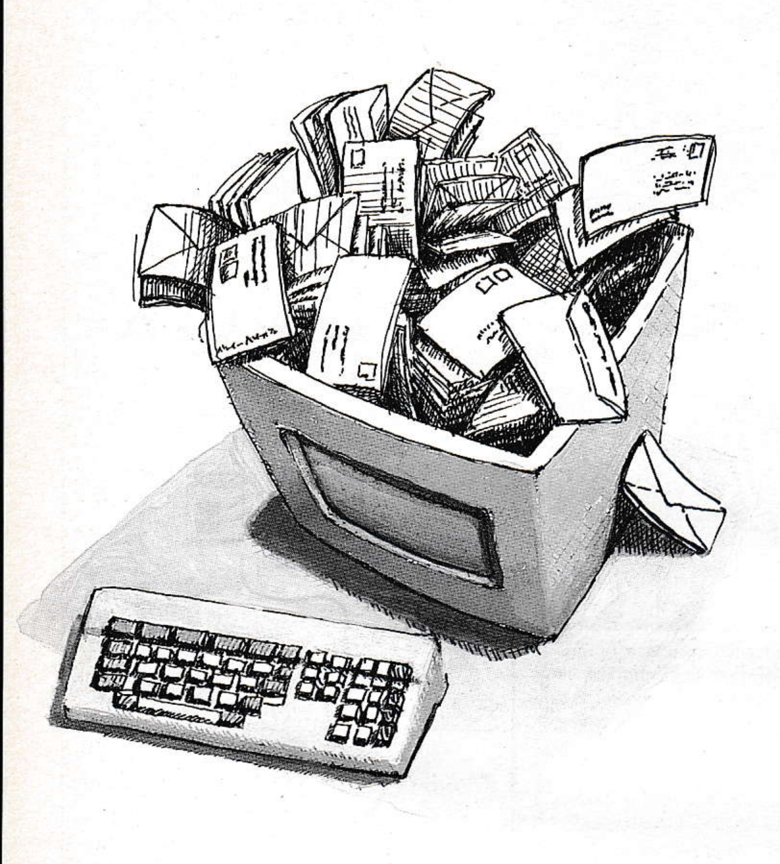
Das Programm hat auf allen ST und TT seinen Dienst klaglos verrichtet und gute Druckergebnisse geliefert. Man darf aber über Einschränkungen im ACC-Betrieb (siehe oben) nicht hinwegsehen. Auch könnte die sogenannte Oberfläche ein gehöriges Facelifting vertragen, und die unechte Menüleiste sollte moderneren Symbolen (Icons) Platz machen.

Das Handbuch darf mit seinen 90 Seiten als angenehm ausführlich bezeichnet werden. Es ist in einen Referenz- und einen Tutorial-Teil aufgespalten. Viele Bilder hat es nicht, weil sich das Hauptinteresse ohnehin nur auf die Oberfläche konzentriert. Auch bei kleinen Programmen macht sich der Umweltschutz allmählich breit, denn es kommt immer mehr nicht so sehr auf aufwendig poppige Verpackung, sondern auf den Inhalt an. So hat Dr. Ackermann auf das übliche PVC-Ringbuch verzichtet und stattdessen eines aus Polypropylen benutzt. Ein nachahmenswerter Gesichtspunkt. Alles in allen kann man DM 78 als Kaufpreis für dieses umfangreiche Druck-Utility nebst Zeichensatzeditor auf jeden Fall akzeptieren.

DK

Bezugsquelle:

Entwicklungsbüro
Dr. Ackermann
Kanalweg 1a
W-8048 Haimhausen
Tel.: (08133)1053



Quelltexte in der ST-Computer

Ich lese die ST-COMPUTER seit 1987 und stelle fest, daß die Programme immer interessanter und besser werden, vor allem für den ATARI TT. Die Assembler-Listings z.B. von Uwe Seimet kann man ja noch abtippen, aber die C-Quelltexte für die CPX-Module sind mir dann doch zu umfangreich.

Wir haben nun an der Uni Saarbrücken den Versuch gemacht, die Quelltexte mit Scanner und Schrifterkennung zu erfassen, leider aber nur mit mäßigem Erfolg. Die Schrift im Heft ist viel zu klein, der farbige Hintergrund stört enorm, und Kopieren mit Vergrößern verschlechtert nur den Kontrast. Natürlich wissen wir, daß Sie einen Diskettenservice anbieten, aber ist es nicht möglich, die Quelltexte in OCR-Schrift abzudrucken?

B. Stumpf, Uni Saarbrücken

Red.: Dieses Problem beschäftigt zahlreiche Leser, und wir haben lange einen Ausweg überlegt. Um es vorwegzunehmen, uns ist keine sinnvolle Alternative eingefallen. Das Abdrucken von Listings war schon immer mit unzähligen Kompromissen begleitet: Wir sind bei jedem Heft aufs neue bemüht, so viele interessante Listings wie möglich abzudrucken. Bei einer größeren Schrift müßten wir entweder den Seitenumfang erweitern (wobei andere Rubriken zu kurz kämen) oder die Anzahl der verschiedenen Themen einschränken (was die Programmierer wenig freuen dürfte). Der farbige Hindergrund hat den Sinn, die Listing vom übrigen Text abzuheben und außerdem nicht zu sehr als Schwarzweiß-Wüste erscheinen zu lassen (darauf könnte man ggf. verzichten). Dann gibt es auch gute Gründe für das Diskettenabonnement. Dort finden Sie - sofern sinnvoll - zu jedem Quelltext auch ein ablauffähiges Programm.

Modem-Port des Mega-STE

Da bei ATARI offenbar niemand weiß, welche Adresse der Modem-2-Port des Mega-STE hat bzw. wie man ihn unter GFA-BASIC anspricht, erlaube ich mir, Ihnen diese Frage zu stellen.

K. Fritz, Bargteheide

Red.: In den Ausgaben September bis November 1991 der ST-COMPUTER berichteten wir ausführlich über die neuen Schnittstellen der Mega-STE- bzw. TT-Computer. In dieser Grundlagen-Reihe, die unter dem Titel „Schnittstellen-Dschungel" lief, sind alle Hardware-Adressen der neuen Ports sowie deren Unterstützung per Software beschrieben. Dort wird zwar in C programmiert, aber anhand der Adressen dürfte die Umsetzung in BASIC kein Problem darstellen.

*

Unsterblicher Ordner

Ich besitze einen ATARI ST mit Festplatte. Nachdem ich einige Disketten auf eine (bis dahin leere) Partition der Festplatte kopiert hatte, konnte ich von dieser Partition bestimmte Ordner nicht mehr löschen. Ich bekam immer die Meldung: „Datei (name) kann nicht gelöscht werden". Dies ist doch sehr ärgerlich, da dadurch wertvoller Speicherplatz verschenkt wird. Wie kann ich diese Ordner wieder entfernen?

Ernst-Jürgen Peters, Essen

Red.: Das Problem ist nicht ganz einfach zu lösen. Bei diesen fraglichen Ordnern ist das sogenannte Schreibschutz-Bit wohl aus Versehen gesetzt worden. Entfernen läßt es sich leider nicht über den Desktop (wie dies z.B. bei Dateien machbar wäre) und auch nicht über die GEMDOS-Funktion

Ein Wort in eigener Sache

In den Jahren, die unsere Zeitschrift existiert, haben wir immer wieder versucht, durch die Beantwortung der bei uns eingehenden Briefe ein wenig Licht in das Dunkel zu bringen, das bei der Arbeit mit dem ATARI ST schon so manch einen aus der Fassung bringen konnte - eine Tatsache, die nicht nur Ihnen, verehrter Leser, sondern auch uns oft genug zu schaffen machte. Nichtsdestotrotz haben wir uns bemüht, die Probleme zu lösen und diverse Leserbriefe zu veröffentlichen, da wir der Meinung waren, daß die jeweilige Thematik auch einen größeren Leserkreis interessieren könnte. Trotzdem gibt es immer wieder Briefe, die wir nicht beantworten können oder dürfen. Damit Sie nicht allzusehr enttäuscht zu sein brauchen oder keine Antwort erhalten, möchten wir Sie bitten, sich an folgende Spielregeln zu halten, die sich aus unserer Erfahrung ergeben haben. Fällt Ihr Brief nicht unter die folgenden Kriterien, hat er gute Chancen, positiv beantwortet oder wenigstens als Hilferuf an unsere Leserschaft gedruckt zu werden.

1. Leider gehen immer wieder Briefe mit dem Wunsch ein, ein Produkt für diesen oder jenen Anwendungsfall vorzuschlagen, verschiedene Produkte bezüglich der Vor- und Nachteile gegeneinander abzuwägen und zu bewerten. Es ist uns aus Wettbewerbsgründen nicht erlaubt, ein bestimmtes Produkt zu favorisieren, selbst wenn wir das eine oder andere in der Redaktion überzeugt einsetzen. Wir können Sie in diesem Fall ausschließlich auf die von uns möglichst objektiven Tests und eventuell anstehende Fachmessen hinweisen. Bedenken Sie bitte, daß auch wir nicht jede Textverarbeitung, jedes Malprogramm und so weiter kennen und bestimmte Produkte dadurch in das Abseits drängen würden.

2. Oft erreichen uns Briefe, die sich positiv oder auch negativ über bestimmte Händler, Softwarehäuser oder deren Produkte auslassen. Sicherlich interessieren uns solche Bemerkungen. Bitte haben Sie aber Verständnis, daß wir weder Lob noch Tadel abdrucken dürfen, da diese Aussagen meist subjektiv sind. Anders sieht die Sache beispielsweise bei Gerichtsurteilen aus, die Sie, verehrte(r) Leser(in), erfochten haben.

3. Aufgrund der Vielzahl an Briefen, die uns täglich erreichen, sind wir leider nicht in der Lage, Programmfehler anhand von Listings oder ähnlichem zu korrigieren. Dennoch sollte ein Problem möglichst detailliert beschrieben sein, denn Ferndiagnosen sind prinzipiell sehr schwer, jedoch mit genauerer Angabe der Symptome eventuell durchführbar.

4. Von Zeit zu Zeit erreichen uns Briefe mit der Bitte, die Adresse des Lesers zwecks allgemeiner Kontaktaufnahme zu veröffentlichen. Würden wir dies in die Tat umsetzen, würde sich der Umfang des anderen redaktionellen Teils beträchtlich verkleinern. Ausnahmen stellen Leser in fernen Ländern dar, für die eine Kontaktaufnahme im eigenen Land recht schwierig ist.

Zum Schluß sollen ein paar Tips eventuell voreilig geschriebene Briefe verhindern.

1. Wenn Sie ein Problem bezüglich einer bestimmten Problematik haben oder an einem bestimmten Produkt interessiert sind, finden Sie interessante Artikel darüber eventuell in vorhergehenden Ausgaben userer Zeitschrift. Zur Auswahl eignet sich das Jahresinhaltsverzeichnis besonders gut, das immer am Jahresende in der ST Computer abgedruckt wird.

2. Sollten die Probleme mit der Handhabung eines Produktes zu tun haben, wenden Sie sich zunächst an Ihren Händler und über diesen an den Distributor beziehungsweise an das Software-Haus. Die Wahrscheinlichkeit, daß Ihnen das Software-Haus weiterhelfen kann, ist um ein Vielfaches höher als die, daß wir Ihnen helfen können.

3. Lesen Sie aufmerksam die Leserbrief-Seite. Viele Fragen wiederholen sich immer wieder, obwohl wir bestimmte Probleme schon mehrfach angesprochen haben.

›nnen wachsen die Ansprüche. OMIKRON.

Ea… Base DAS FLEXIBL… DATENBANKSYSTEM

»Sehr gutes Datenbanksystem für Anfänger und Profis« (ST-Magazin 8/90)

248,– *

…schreib-
…FE

…hlechte Zeiten …ür Fehlerteufel! Rechtschreibprüfer für Calamus, Tempus etc.



99,– *

MORTIMER PLUS

Wohl dem, der einen Butler hat! Vielseitiges Multi-Utility.

»Mortimer, über-nehmen Sie!«

129,– *



BASIC COMPILER 3.5

Der neue Compiler. Nutzt FPU, arbeitet mit Großbildschirmen, erzeugt TT-Lauffähiges.

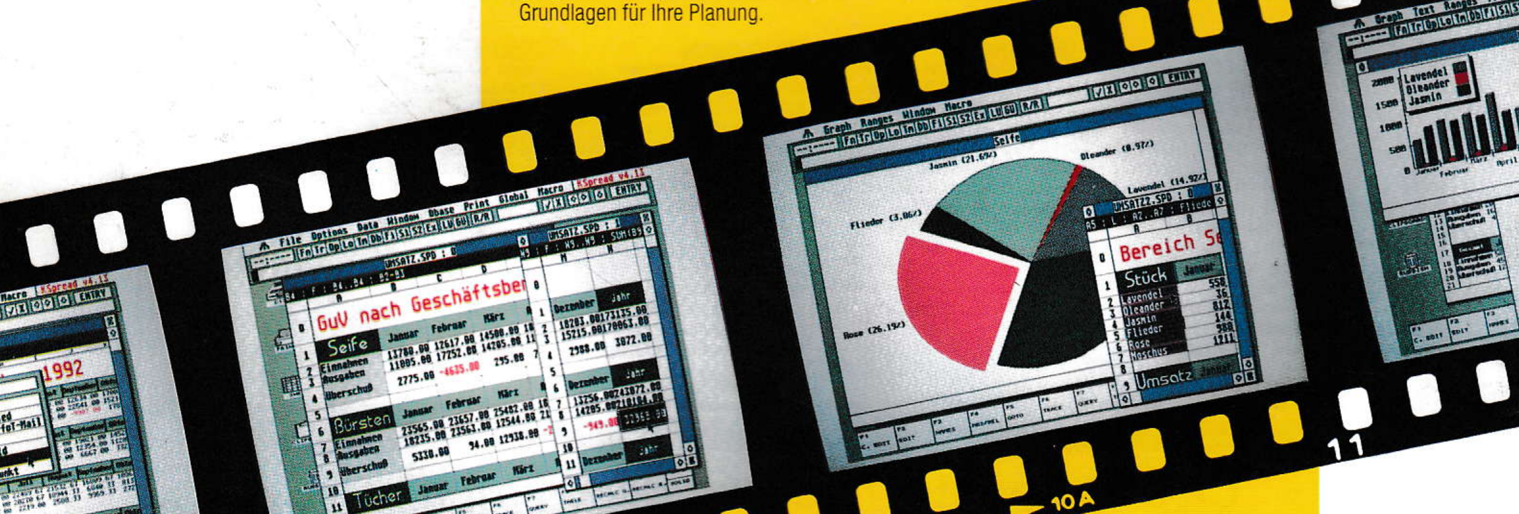
229,– *



TABELLENKALKULATION SCHAFFT DURCHBLICK

Eine Tabellenkalkulation ist ein elektronisches Rechenblatt. Sie geben alle Ausgangsdaten und die Zusammenhänge ein, z.B. Umsatz = Stückzahl × Preis. Wenn Sie jetzt das Feld mit dem Preis ändern, berechnet K-Spread 4 automatisch den dadurch veränderten Umsatz.
Da auch umfangreiche Rechenblätter so in Sekundenschnelle durchgerechnet werden, schaffen »was..wäre.. wenn«-Überlegungen mit K-Spread 4 schnell fundierte Grundlagen für Ihre Planung.

Die Benutzeroberfläche ist konsequent GEM-Standard. Bei vielen Funktionen kommen Sie so von alleine darauf, wie sie funktionieren.
Als einzige Tabellenkalkulation auf dem ST arbeitet K-Spread 4 mit bis zu acht Fenstern. Durch »herüberziehen« können Sie somit blitzschnell Daten zwischen verschiedenen Rechenblättern übertragen.



K-Spread 4 läuft auf allen Atari ST und TT (ab 1 MB). Es unterstützt sogar Farbe und Großbildschirme. Wie Sie sehen, wird die Farbe auch genutzt. Präsentations-Grafiken werden in Farbe viel klarer; und negative Zahlen kann K-Spread automatisch rot darstellen.

K-Spread ist seit Jahren Marktführer in England. Über 10.000 Stück wurden dort verkauft. Die deutsche Version wurde von OMIKRON für deutsche Erfordernisse maßgeschneidert. K-Spread 4 erhalten Sie bei allen OMIKRON.Vertragshändlern oder direkt bei OMIKRON. * Unverbindliche Preisempfehlung

DM 248,–*

OMIKRON.Soft- + Hardware GmbH
Sponheimstr. 12a · D-7530 Pforzheim
Telefon 0 72 31/35 60 33



XEST, Hirschengasse 8, A-1060 Wien
OMIKRON, France, 11, rue dérodé, F-51100 Reims
DTZ DataTrade AG, Landstr. 1, CH-5415 Rieden/Baden
Jotka Computing, Postbus 8183, NL-6710 AD Ede